НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»

Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
"___" червня 2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки ***6.050102 «Комп'ютерна інженерія»***
на тему: Програмні засоби аналізу звукових сигналів з використанням нейронних мереж

Виконав: студент IV курсу, групи КВ-53
Герасимюк Владислав Анатолійович
(підпис)

Керівник_____
_____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М.

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____
_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

Герасимюку Владиславу Анатолійовичу

1. Тема проекту «Програмні засоби аналізу звукових сигналів з використанням нейронних мереж»,

керівник роботи _____

_____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом роботи

_____

Вихідні дані до роботи: див. Технічне завдання.

4. Зміст пояснювальної записки: див. Пояснювальна записка

5. Перелік графічного матеріалу:

- _____ Презентація
- _____ Схема алгоритму передачі десеріалізованих аудіо та запуск нейронної мережі.

• _____ Схема алгоритму попередньої обробки звукового сигналу.
• _____ Структурна схема рекурентної нейронної мережі
• _____ Структурна згорткової рекурентної нейронної мережі

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів
1.	Вивчення літератури за тематикою роботи	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломної роботи	10.05.2019
5.	Підготовка матеріалів другого розділу дипломної роботи	18.05.2019
6.	Підготовка графічної частини дипломної роботи	20.05.2019
7.	Оформлення документації дипломної роботи	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Студент _____ (підпис) _____ (ініціали, прізвище)

Керівник проекту _____ (підпис) _____ (ініціали, прізвище)

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045480.002 ТЗ	Програмні засоби аналізу	4		
			звукових сигналів з використанням			
			нейронних мереж.			
			Технічне завдання			
	А4	ІАЛЦ.045480.003 ТП	Програмні засоби аналізу	2		
			звукових сигналів з використанням			
			нейронних мереж.			
			Відомість технічного завданння			
	А4	ІАЛЦ.045480.004 ПЗ	Програмні засоби аналізу	62		
			звукових сигналів з використанням			
			нейронних мереж.			
			Пояснювальна записка			
	А4	ІАЛЦ.045480.005 Д1	Програмні засоби аналізу	1		
			звукових сигналів з використанням			
			нейронних мереж.			
			Попередньої обробки звукового			
			сигналу.			
			Схема Алгоритму			

					ІАЛЦ.045480.001 ОА
Зм	*Лист*	*№ докум.*	*Підп*	*Дат*	
Розроб.		Герасимюк В.А			Програмні засоби аналізу звукових сигналів з використанням нейронних мереж ***Опис альбому***
Перев.		Петрашенко А.В.			
Н. контр.		Клятченко Я.М.			
Затв.		Тарасенко В.П.			

Літ.	*Лист*	*Листі*
	1	2

КПІ ім. Ігоря Сікорського, ФПМ, КВ-53

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.006 Д2	Програмні засоби аналізу	1		
			звукових сигналів з використанням			
			нейронних мереж.			
			Передачі десеріалізованих аудіо			
			та запуск нейронної мережі			
			Схема алгоритму			
	А4	ІАЛЦ.045480.007 Д3	Програмні засоби аналізу	1		
			звукових сигналів з використанням			
			нейронних мереж.			
			Рекурентної нейронної мережі			
			Структурна схема			
	А4	ІАЛЦ.045480.008 Д3	Програмні засоби аналізу	1		
			звукових сигналів з використанням			
			нейронних мереж.			
			Згорткової нейронної мережі			
			Структурна схема			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045480.001 ОА	Арк. 2

Зміст

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування роботи – «Програмні засоби розпізнавання аудіо за допомогою нейронних мереж».

Область застосування: Автоматизація та покращення систем розпізнавання звукових сигналів.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр спеціалізованих комп'ютерних систем», затверджене кафедрою спеціалізованих комп'ютерних систем

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення системи для розпізнавання та класифікації звукових сигналів методами нейронних мереж мовою програмування Python.

4. ДЖЕРЕЛА РОБОТИ

Джерелами роботи є науково-технічна література по методам вилучення функції з аудіо, алгоритмам глибоких нейронних мереж, електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до системи, що розробляється:

- Візуалізація архітектури нейронних мереж, що використовуються;
- Отримання результату роботи нейронної мережі, а саме ймовірності належності поданого на вхід класу, до кожного з 10 запропонованих.

5.2. Вимоги до апаратного забезпечення:

Комп'ютер на базі процесора сімейства AMD або сімейства Intel, з оперативною пам'яттю 4096 Мбайт і більше, дискретною відеокартою сімейства Nvidia, з відеопам’яттю 1024 Мбайт або вище.

5.3. Вимоги до програмного забезпечення:

1. Операційна система одного з сімейств:
    - Windows 7 або вище;
    - Linux 14.04 або вище;
    - Mac OS X 10.7.3 (Lion) або вище.
2. Встановлене програмне забезпечення Nvidia Cuda 8.2 або вище.
3. Встановлене програмне забезпечення фреймворку для глибокого навчання Pytorch 0.4 або вище, Keras 1.10 або вище.s
4. Додаткові інсталяційні пакети.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

№ п/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документація загальна		
			Новорозроблена		
1	А4	ІАЛЦ.045480.004 ПЗ	Програмні засоби аналізу звукових	62	
			сигналів з використанням		
			нейронних мереж.		
			Пояснювальна записка роботи.		
2	А4	ІАЛЦ.045480.005 Д1	Програмні засоби аналізу звукових	1	
			Сигналів з використанням		
			нейронних мереж.		
			Попередньої обробки звукового		
			сигналу.		
			Схема Алгоритму		
3	А4	ІАЛЦ.045480.006 Д2	Програмні засоби аналізу звукових	1	
			сигналів з використанням		
			нейронних мереж.		
			Передачі десеріалізованих аудіо		
			та запуск нейронної мережі		
			Схема алгоритму		
4	А4	ІАЛЦ.045480.007 Д3	Програмні засоби аналізу звукових	1	
			сигналів з використанням		
			нейронних мереж.		
			Рекурентної нейронної мережі		
			Структурна схема		
5	А4	ІАЛЦ.045480.008 Д4	Програмні засоби аналізу звукових	1	
			сигналів з використанням		
			нейронних мереж.		
			Згорткової нейронної мережі		
			Структурна схема		

Зм.	Арк	№ докум	Підпис	Дата
Розроб.		Герасимюк В.А.		
Перевір.		Петрашенко А.В.		
Н. контр.		Клятченко Я.М.		
Затв.		Тарасенко В.П.		

ІАЛЦ.045480.003 ВП

Програмні засоби аналізу звукових сигналів з використанням нейронних мереж

Відомість роботи

Літ.	*Арк.*	*Аркушів*
	1	1

«КПІ ім. Ігоря Сікорського», ФПМ, КВ-53

ЗМІСТ

Table of Contents

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ЗС – звуковий сигнал.

НМ – нейронна мережа.

ФСК - функція спектрального контрасту.

AI – штучний інтелект.

Big Data- великі данні.

Black box- чорна скриня.

Cell state - Стан комірки.

CNN- згорткові нейронні мережі.

Dataset - набір даних

DCT – дискретне косинусне перетворення.

Dilated causal convolutions – поширювальні причинні згортки.

Feed-forward neural networks – мережі прямого поширення.

FFT – швидке перетворення Фур’є.

Forget gate layer - шар фільтру забування.

Input layer gate – стан вхідного фільтру.

Long short-term memory - Довга короткострокова пам'ять.

Machine learning - машинне навчання.

MFCC - Мел-частотні кепстральні коефіцієнти.

RNN - Рекурентні нейронні мережі.

STFT - короткочасне дискретне перетворення Фур’є.

Time series – часові ряди.

ВСТУП

Останні кілька років у світі стрімко розвивається сфера машинного навчання, нейронних мереж та штучного інтелекту. Невеликі компанії та масштабні корпорації прагнуть впровадити штучний інтелект у виробництво, в будь-якій сфері бізнесу тільки й чутно про штучний інтелект.

Існує кілька факторів, які спричинили таку популярність методів машинного навчання. Основний – машина працює краще ніж людина, машину легше навчити, вона здійснює менше помилок, у неї відсутній людський фактор. Ера автоматизації настала вже давно, зараз вона на піку свого розвитку. Big data- світ великих даних, кожного дня створюється 2.5 квінтільйона байтів даних [1], а аналізується з цих даних менше ніж 1% [2]. Саме тому потрібно використовувати інформацію, яка існує, в правильному напрямку- для блага суспільства та розвитку технологій.

Відтак, штучний інтелект - це шлях до вирішення багатьох проблем суспільства, від росту економіки до зниження корупції. Однією з цілей даної дипломної роботи є, показати, що автоматизація і штучний інтелект є сильним інструментом в сучасному світі.

Завданням дипломної роботи – є побудова системи штучного інтелекту, яка, за допомогою нейронних мереж буде розпізнавати та класифікувати звукові сигнали. Така систему може так чи інакше використовуватись у таких сферах як, розпізнавання людської мови, робототехніка, сенсорні системи захисту, кібербезпека та інші.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОЇ РОБОТИ

1.1. Підходи до кодування звукових сигналів та вилучення функцій, що характеризують унікальність звукового сигналу

Звуковий сигнал – це представлення звуку серією двійкових чисел.

Спектральний аналіз звуку - метод для встановлення акустичної структури звуків, представляють собою акустичний сигнал, що безперервно змінюється у часі, який утворюється рядом частотних складових з різною інтенсивністю.

Задля реалізації розпізнавання, обробки та класифікації ЗС крім алгоритму розпізнавання, важливою частиною є попередня обробка даних та вилучення ознак з аудіо.

У роботі розглядаються такі підходи до обробки та вилучення ознак як:

- Мел-частотні кепстральні коефіцієнти – представлення лог потужності спектра в мел-частотній області. Описують потужність огинаючої спектра, що характеризує модель мовного тракту.
- Спектральний центроїд – дозволяє ефективно визначати зашумленні ділянки звукового сигналу.
- Хроматограма - потужний метод вилученні інформації з аудіо, суть якого полягає в проеціюванні спектру на 12 різних напівтонів.
- Спектральний контраст – враховує спектральну долю та спектральний пік, обчислює їх різницю у кожному частотному діапазоні.
- Темпорограма – локальна автокореляція огинаючої початку(характеристика ритму).

Описані підходи далі використовуються безпосередньо для вилучення ознак.

Так як основним інструментом машинного навчання було обрано нейронні мережі, а саме ті архітектури, які дозволяють працювати з time series, вони потребують відповідного подання та обробки звукових сигналів.

Однією з основних проблем, яка потребує вирішення є те, що ЗС різної довжини, відтак їх потрібно нормалізувати та вирівняти по часу, задля досягнення кращої якості класифікації.

Важливою властивістю нейронних мереж є те, що вони дозволяють автоматизувати процес агрегації та виділення важливих частин з сигналу, це означає, що нейронна мережа є black box, але завдяки методу зворотнього поширення помилки, алгоритм вивчить і виділить що є важливим, а що ні.

Інженерною задачею в даному випадку є те, щоб 'допомогти' нейронній мережі правильно вивчити дані, для цього в роботі було використано короткочасне дискретне перетворення Фур'є(STFT)[3].

STFT дозволяє моделювати часовий ряд таким чином, щоб всі навчальні приклади були фіксованої заданої довжини.

Таким чином, до всього набору звукових сигналів було застосовано перетворення Фур'є, після цього емпіричним методом було підібрано чотири підходи до вилучення корисних функції з перетворених аудіо, а саме:

1) Мел-частотні кепстральні коефіцієнти

2) Спектральний центроїд

3) Хроматограма

4) Спектральний контраст

Було виявлено, що темпорограма, яка є характеристикою ритму, не є корисною для вирішення поставленої задачі і не дає жодної корисної інформації.

В підсумку було отримано навчальний набір даних, який складався оброблених звукових сигналів розмірністю (l, s, f) , де l – кількість

прикладів для навчання(звукових сигналів), s- довжина часової серії, що була підібрана емпіричним способом у перетворенні Фур'є, f- кількість ознак, вилучених з аудіо сигналів.

1.2. Аналіз існуючих видів нейронних мереж та огляд архітектур для вирішення задачі

Штучна нейронна мережа – є математичною моделлю, а саме системою взаємодіючих між собою та з'єднаних штучних нейронів(процесорів). Кожен з процесорів приймає та періодично передає сигнали іншим процесорам, взаємодіючи між собою нейрони складають велику мережу з керованою взаємодією та властивістю навчатись в процесі роботи.

Задача класифікації звукових сигналів є трудомісткою для її якісного вирішення, для того, щоб алгоритм навчився правильно класифікувати звуковий сигнал, потрібно великий обсяг навчальних даних, в роботі використано dataset об'ємом 8 гб., саме тому було вирішено не брати до уваги класичні алгоритми машинного навчання, що показують гарні результати на інших задачах з невеликими обсягами інформації, але є неефективними та погано узагальнюючими у big data.

В даному розділі розглянуто лише глибокі нейронні мережі зі зворотнім зв'язком в контексті роботи з time series.

Коли йде мова про роботу с часовими рядами, будь-якому machine learning спеціалісту в першу чергу спадають на думку рекурентні нейронні мережі.

Рекурентні нейронні мережі – тип нейронних мереж, що використовується переважно в роботі з послідовностями, включаючи як часові послідовності так і текстові. Основною ідеєю RNN є те, що вони використовують інформацію послідовно, якщо мова про речення- з початку

до кінця, якщо мова про часову серію- по часу. В класичних нейронних мережах всі входи і виходи незалежні один від одного, але для багатьох задач це не підходить. Якщо потрібно передбачити наступний момент часу- потрібно щоб мережа мала інформацію про минулі моменти. RNN мають назву рекурентні, тому що виконується одна й та ж задача для кожного елементу послідовності. А вихід такої мережі залежить від попередніх обчислень[4].

В цілому, рекурентні нейронні мережі є основним та сильним інструментом у роботі з часовими рядами. Але через те, що мережі такого типу вимагають значних апаратних ресурсів для навчання, було вирішено розглянути й інші архітектури нейронних мереж.

У світі машинного навчання все розвивається надзвичайно стрімко, існує багато парадоксів, одним з них є застосування згорткових нейронних мереж, що призначені для роботи з зображеннями, у задачах с часовими рядами.

Слід сказати, що CNN досить добре показали себе у задачах пов'язаних з розпізнаванням аудіо, тексту, голосу. Це довели вченні з корпорації Google[5].

Згорткова нейронна мережа – архітектура штучних нейронних мереж прямого поширення, під прямим поширенням мається на увазі, що нейрони(процесори) розбиті на групи, які називаються шарами. Коли такий тип мережі опрацьовує дані, то активація значень змінних рахується послідовно, активація відбувається послідовно- від першого шару до останнього, він є виходом нейронної мережі, що вказує результат. Важливим є те, що активації всередині одного шару рахуються незалежно один від одного, саме тому їх модна обчислювати паралельно - за допомогою сучасних графічних процесорів, що робить CNN дуже потужним інструментом.

Як було описано вище, CNN початково призначались та були створені для роботи з зображеннями, саме тому класичними згортковими

мережами рахується 2D згортання, тобто мережа потребує на вхід зображення та згортає його відповідно.

Для вирішення задачі класифікації аудіо сигналу, було використано інструмент, що називається Temporal Convolutional Neural Networks[6]- це підвид CNN, що дозволяє обробляти time series. В основі нейронних мереж цього типу лежить 1D згортання ‘вперед’, тобто в процесі згортання інформація передається спочатку у кінець по довжині часової серії, при цьому ознаки додаються чи діляться між собою. Згортання ‘вперед’ забезпечують пропуски між нейронами, тобто на кожному шарі роботи мережі, пропускається якась кількість нейронів- інформація передається вперед.

У даній дипломній роботі розглянуто, використано та порівняно результати двох архітектур нейронних мереж для опрацювання часових рядів, було вирішено не брати до уваги інші поглиблені архітектури, яких існує дуже багато, тому що для досягнення мети роботи вистачить таких сильних інструментів як RNN та CNN.

1.3. Обґрунтування теми дипломної роботи

У даній дипломній роботі є дві мети- теоретична та практична. Теоретичною метою є дослідження способів, підходів та аналіз засобів що в подальшому дозволять підібрати ту комбінацію інструментів, що дозволять максимізувати результат конкретної задачі в залежності від факторів та цілей.

Практичною метою є вирішення реальної задачі з життя- розпізнавання та класифікація звукових сигналів. В даній роботі не наголошено, що контекстом аудіо сигналів є звуки міста, з відкритого набору данних[7]. Це зроблено спеціально, адже початково стояла підзадача- розробити систему, яку можна буде з легкістю переносити на інші контексти, застосовувати для інших задач. Відтак, міняючи параметри

навчання, функції втрат, можна переносити дану систему на задачі класифікації інших об’єктів, регресії, генерації звуку та інші.

Як було сказано, основним інструментом для досягнення мети було обрано нейронні мережі, перевагами такого підходу є:

1) Можливість паралельного навчання, що в рази прискорює швидкість навчання та донавчання.
2) Властивість нейронних мереж до узагальнення, тобто нейронна мережа навчає сама себе і узагальнює знання на приклади, яких не знає, що дозволяє вирішувати масштабні задачі.
3) Стійкість до шуму у вхідних даних.
4) Адаптивність до зміни даних.

Таким чином вирішено, що нейронні мережі є найкращим інструментом для подібного роду задач та розробці систем.

1.4. Обґрунтування вибору середовища розробки

Коли метою є побудова адаптивної, стійкої системи штучного інтелекту- мова йде так чи інакше про Big Data, а там де є великі дані, там є затрати ресурсів та часу.

Так як було встановлено, що класичні методи машинного навчання є неефективнимим у вирішенні задачі, єдиним інструментом залишаються нейронні мережі. Недоліком будь-якого об’ємного механізму є ресурсо- та часо- затратність. Інженерною задачею в цьому випадку є вибір бібліотек та фреймворків машинного навчання, що забезпечать ефективний доступ до графічного процесору комп’ютера, дозволять ефективно використовувати апаратні потужності.

Було проаналізовано всі основні фреймворки для глибокого та машинного навчання, порівняно їхні недоліки та переваги, було вибрано передову розробку від facebook [7]- PyTorch[8].

PyTorch – сучасна бібліотека глибокого навчання, з відкритим вихідним

кодом, створена на базі Torch[9].

Бібліотека представляється двома основними високорівневими моделями:

1) Глибокі нейронні мережі
2) Тензорні обчислення з можливістю прискорення та розпаралелювання завдяки графічному процесору.

Основною відмінністю PyTorch від аналогів є те, що вона має динамічний граф обчислення, що дає максимальну гнучкість та можливість розширювати архітектуру, дозволяє використовувати в обчисленнях всі можливості мови програмування, що використовується.

Мовою програмування було обрано Python.

2 ОПИС ОБРАНИХ ФУНКЦІЙ ТА МЕТОДІВ ВИЛУЧЕННЯ ОЗНАК ЗІ ЗВУКОВОГО СИГНАЛУ

2.1. Опис методів

Практична частина дипломної роботи умовно розділяється на дві частини, першою є обробка та вилучення функції зі звукових сигналів, другою – передача вилучених ознак на алгоритм машинного навчання та подальше навчання алгоритму.

Виділяється два основних підходи до аналізу аудіо сигналів:

1) Ритмічний аналіз(аналіз темпу).
2) Спектральний аналіз.

В даній роботі було досліджено кожен з підходів та функції властиві кожному з них.

Аналіз темпу – витяг локального темпу та інформації про ритм із звукових сигналів є важкою задачею. Було проведено дослідження, які доводять, що в аудіо сигналах містяться різні рівні темпу, такі як міра, такт, тактум, що значно усладнюють витят абсолютного темпу. У статті[8] було представлено метод, що дозволяє кодувати інформацію про локальний темп. Вченими було введено концепцію циклічних темпограм, які є аналогією до концепції циклічних хрома-характеристик. Також в наведеній статті описано метод, який дозволяє отримувати циклічні темпрограми з аудіо.

В практичному випадку обчислення темпрограми зводиться до обчислення локальної автокореляції огинаючої сили початку.

Автокореляція – статистична модель взаємозв’язку між послідовностями величин одного ряду, взятих із зсувом, для стохастичного процесу – із зсувом по часу.

Автокореляційна функція – взаємозв’язок між сигналом і його зсунутою

копією від величини часового зсуву.

Автокореляційна функція(АКФ) сигналу $f(t)$ визначається інтегралом(для детермінованих сигналів):

$$\Psi(\tau) = \int_{-\infty}^{\infty} f(t)f^*(t-\tau)dt \quad (2.1)$$

Та показує зв'язок сигналу з копією самої себе, що є зміщеною на величину τ. Зірочка у формулі означає комплексне сполучення.

Для стохастичних процесів АКФ випадкової функції $X(t)$ має вигляд:

$$K(\tau = \mathbb{E}\{X(t)X^*(t-\tau)\} \quad (2.2)$$

, де E математичне очікування зірочка означає комплексну зміну.

Автокореляційна функція є потужним методом в математичному моделюванні та аналізі та роботі с часовими рядами.

Обчислення автокореляційної функції виконується за допомогою швидкого перетворення Фур'є та є прямопропорційною елементами послідовності довжини n:

$$\Psi(\tau) \sim Refft^{-1}(|fft(\bar{x})|^2) \quad (2.3)$$

Швидке перетворення Фур'є – алгоритм, що дозволяє прискорено обчислити Дискретне перетворення Фур'є, дозволяє отримати результат менше ніж за $O(N^2)$.

Алгоритм, що дозволяє обчислити темпрограму з аудіо, складається з наступних пунктів:

1) Центруються вікна автокореляції
2) Із часової серії формується фрейм заданої величини
3) Обрізається до довжини початкового сигналу
4) Проходиться вікном по фрейму, застосовується функція автокореляції та нормалізується

На рисунку 2.1 зображено середню та глобальну автокореляцію.

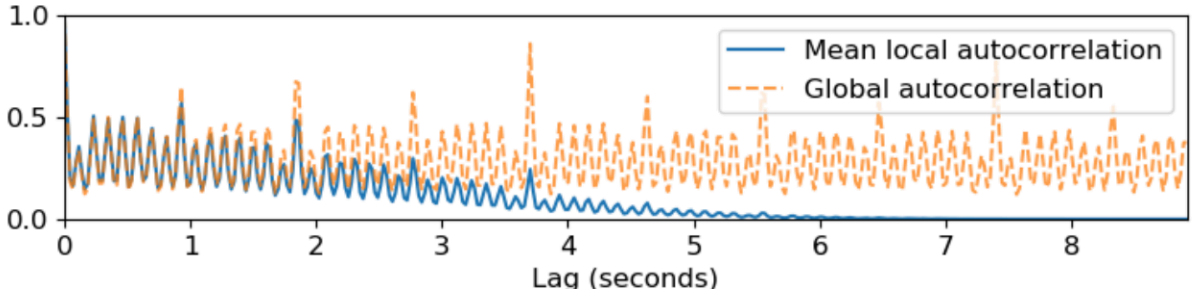

Рисунок 2.1. - Автокореляція

Темпрограма є потужним засобом обробки аудіо та вилучення інформації з нього, даний метод є корисним та діючим при роботі з музичними сигналами та голосом, тому що в таких аудіо міститься чіткий ритм. Аналітично було встановлено, що даний метод не є корисним для вилучення інформації з аудіосигналів, що використано в данній дипломній роботі.

Більше різноманіття методів вилучення інформації з даних описано у спектральному аналізі аудіосигналів.

Було проаналізовано та досліджено всі методи, що описано в пункті 1.1.. Інженерною задачею є правильний підбір корисних методів, які дадуть максимум інформації, та не будуть створювати додаткового шуму для нейронної мережі.

Емпіричним способом було виявлено методи, що відповідають заданим критеріям, а саме:

1) Мел-частотні кепстральні коефіцієнти.
2) Спектральний центроїд.
3) Хроматограма.
4) Спектральний контраст – враховує спектральну долю та спектральний пік, обчислює їх різницю у кожному частотному діапазоні.

Мел-частотні кепстральні коефіцієнти(MFCC).

Мел- є психофізичною одиницею висоти звуку. Оцінка звуку по висоті

засновується на статистичних дослідженнях big data про сприйняття висоти тону звуку.

Формулою, що дозволяє перетворити значення частоти звуку(Гц) у значення висоти(мел) є:

$$m = 1127{,}01048 \ln(1 + f/700) \quad (2.4)$$

Зворотнє перетворення:

$$f = 700(e^{\frac{m}{1127{,}01048}} - 1) \quad (2.5)$$

Графік залежності висоти звуку в мелах від частоти коливань рисунок 1.2

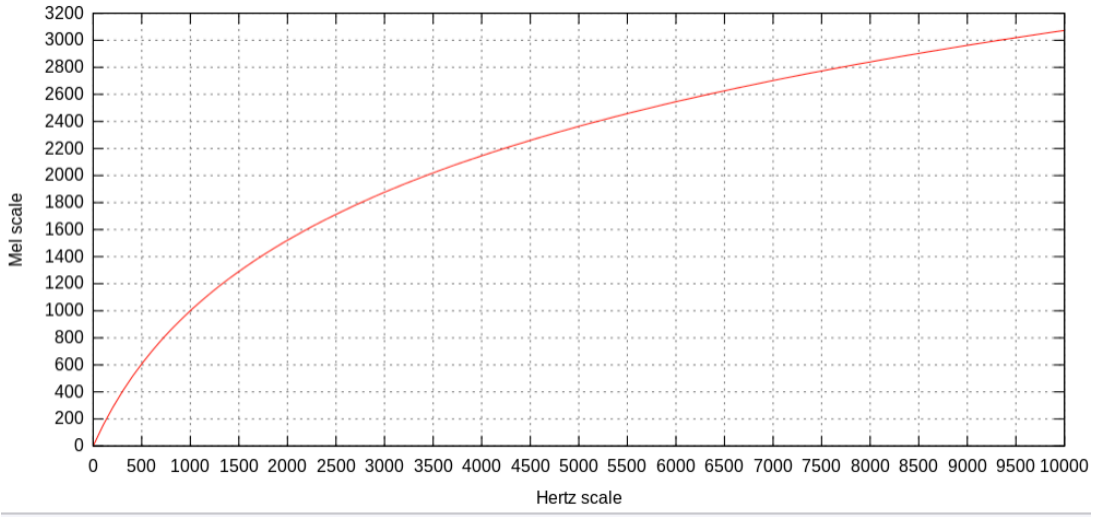

Рисунок 1.2. – Залежність висоти від частоти звуку

Враховуючи те, що аудіо, які використовуються для навчання системи являють собою акустичну хвилю, то її можна представити як сукупність генераторів тонових сигналів, фільтрів та шумів:

1) Генератор імпульсної послідовності тонів.
2) Генератор шумів(випадкової послідовності чисел).
3) Коефіцієнти цифрового фільтру.
4) Нестаціонарний цифровий фільтр.

Сигнал нестаціонарного цифрового фільтру можна подати у вигляді

згортки:

$$f(t) = s(t) \otimes h(t) \quad (2.6)$$

, де s(t)-початковий вигляд звукової хвилі, h(t)-характеристика фільтру.

В частотній області:

$$F(\omega) = S(\omega)H(\omega) \quad (2.7)$$

Отримаємо суму замість множення, прологарифмувавши формулу:

$$\ln [S^2(\omega)H^2(\omega) = lnS^2(\omega) + lnH^2(\omega) \quad (2.8)$$

Далі застосовується перетворення Фур'є для того, щоб перетворити отриману суму так, шоб отримати набори характеристик вихідного сигналу та фільтру, які не перетинаються між собою:

$$C(q) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \ln [F(\omega)]^2 e^{iwq} \, d\omega \quad (2.9)$$

В підсумку, повний алгоритм перетворення аудіо в mfcc виглядає таким чином:

1) Вхідний звуковий сигнал записується у дискретному вигляді

$$x[n], 0 \leq n \geq \text{N} \quad (2.10)$$

2) Застосовується перетворення Фур'є

$$x_a[k] = \sum_{n=0}^{N-1} x[n] e^{\left(-\frac{2mi}{N}\right)kn}, 0 \leq k \geq N \quad (2.11)$$

3) Складається система фільтрів, використовуючи віконну функцію

$$H_m = \begin{cases} 0, \ k < \ f[m-1] \\ \frac{(k - f[m-1])}{(f[m] - f[m-1])}, f[m-1] \ \leq k < f[m] \\ 0, \ k \ > \ f[m+1 \end{cases} \quad (2.12)$$

, де $f[m]$ виходить з рівності

$$f[m] \ = \ (\frac{N}{F_s}) B^{-1}(B(f_1) + m \frac{B(f_h) - B(f_1)}{M+1} \quad (2.13)$$

, де $B(b)$ – перетворення значення частоти в мел-шкалу, тоді

$B^{-1}(b) = 700(exp(b/1125) - 1)$ (2.14)

4) Обчислюється енергія кожного вікна

$S[m] = ln(\sum_{k=0}^{N-1} |X_a[k]|^2 H_m[k]), 0 \leq m < M$ (2.15)

5) Застосовується дискретне косинусне перетворення

$c[n] = \sum_{m=0}^{M-1} S[m] cos(\pi n(m + 1/2)/M, 0 \leq n < M$ (2.16)

Фінальний набір MFCC зображено на рисунку. 1.3.

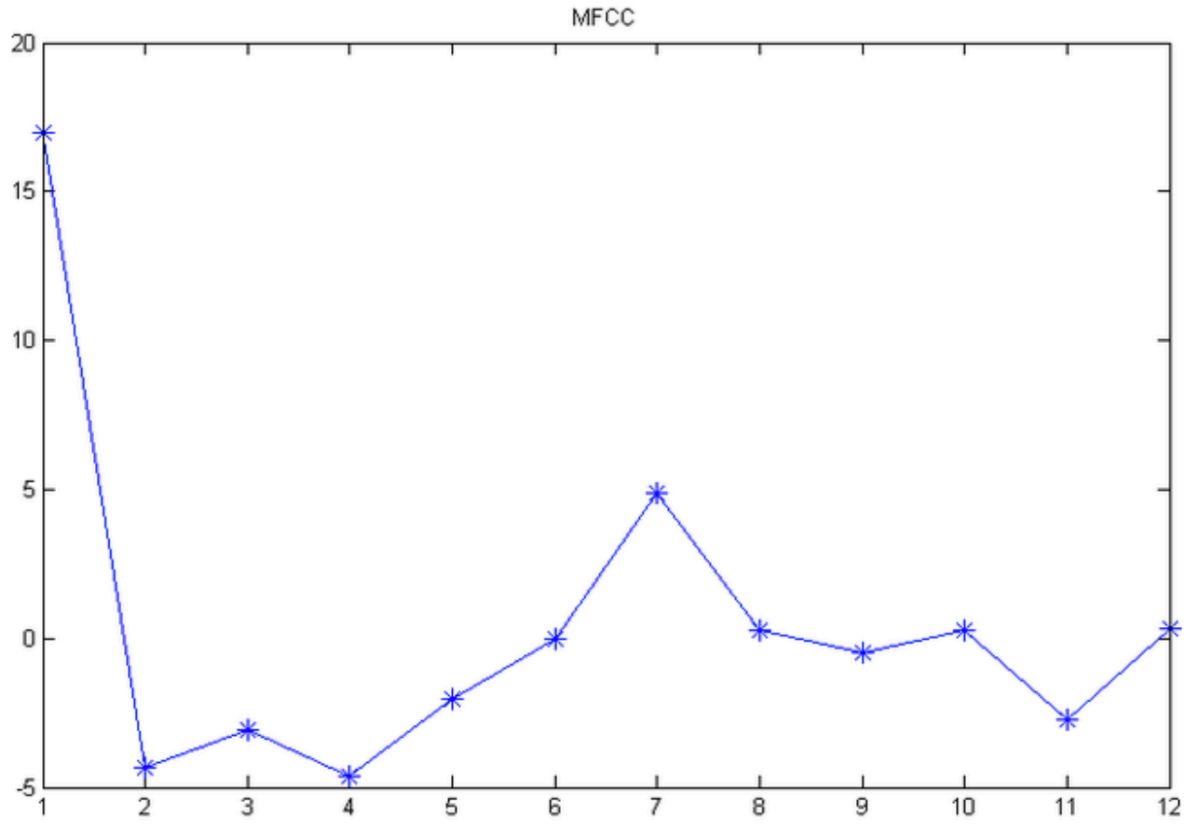

Рисунок 1.3. – Набір MFCC

Спектральний центроїд.

Кожен кадр спектрограми величини нормалізується і розглядається як розподіл по частотним бінам, з кожного з яких витягується центроїд кадру. Спектрограмою є зображення, що показує залежність спектральної густини потужності від часу.

Спектрограма сигналу s(t) може бути оцінена шляхом обчислення квадрату амплітуди віконного перетворення Фур'є сигналу s(t)

$$\text{spectrogram}(t,\omega) = |\text{STFT}(t,\omega)|^2 \quad (2.17)$$

На рисунку 1.4 зображено приклад спектрограми

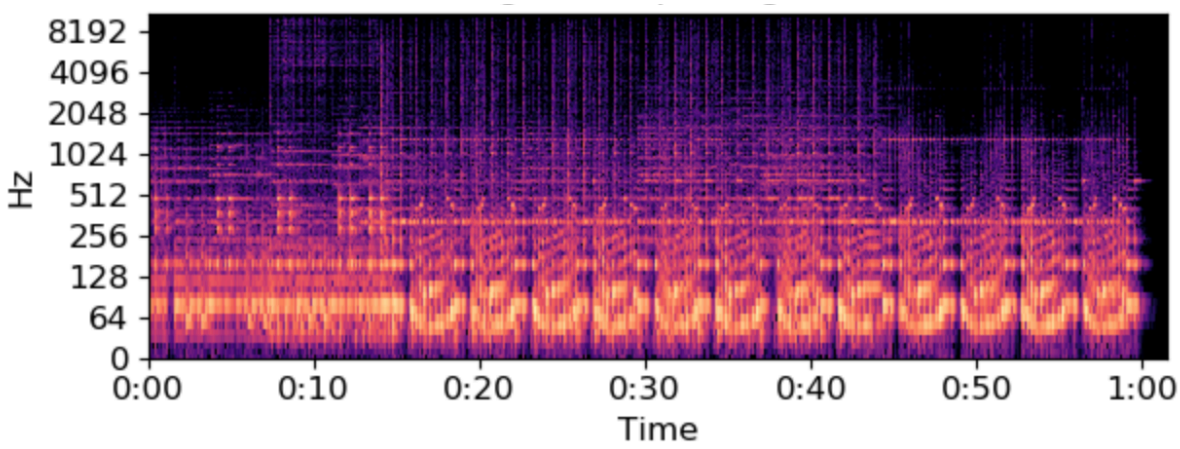

Рисунок 1.4. - Спектрограма

Віконне перетворення Фур'є (Short-time Fourier Transform) – є різновидом перетворення Фур'є, яка обчислюється за:

$F(t,\omega) = \int_{-\infty}^{\infty} f(\tau)W(\tau - t)e^{-i\omega\tau}d\tau$ (2.18)

, де $W(\tau - t)$ – деяка віконна функція.

Типи віконних функцій:

1) Прямокутне вікно. Виходить автоматично при обмеженні вибірки N відліками.

$$w(n) = \begin{cases} 1, & n \in [0, N-1] \\ 0, & n \notin [0, N-1] \end{cases} \quad (2.19)$$

2) Вікно Ханна

$$w(n) = 0.5\ \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \quad (2.20)$$

, де N-ширина вікна, рівень бокових пелюстків – 31.5 дБ.

3) Вікно Хеммінга

$$w(n) = 0.53836 - 0.46164\ \cos\left(\frac{2\pi n}{N-1}\right) \tag{2.21}$$

4) Вікно Блекмана

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right)$$
$$a_0 = \frac{1-\alpha}{2}; \quad a_1 = \frac{1}{2}; \quad a_2 = \frac{\alpha}{2} \tag{2.22}$$

На рисунку 1.5. зображено обчислений спектральний центроїд

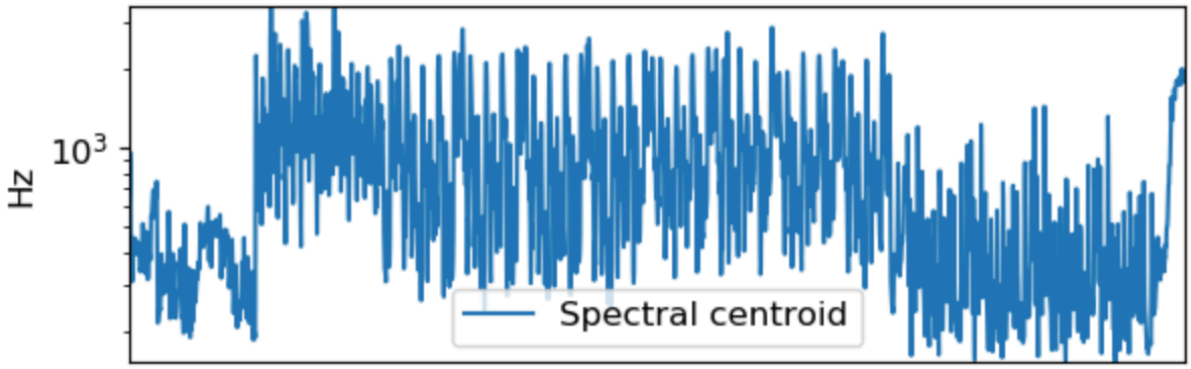

Рисунок 1.5. – Спектральний центроїд

Хроматограма.

Обчислюється хроматограма зі спектрограми потужності.

Хроматограмою є крива, яка показує залежність сигналу від часу.

На рисунку 1.6. зображено хроматограму.

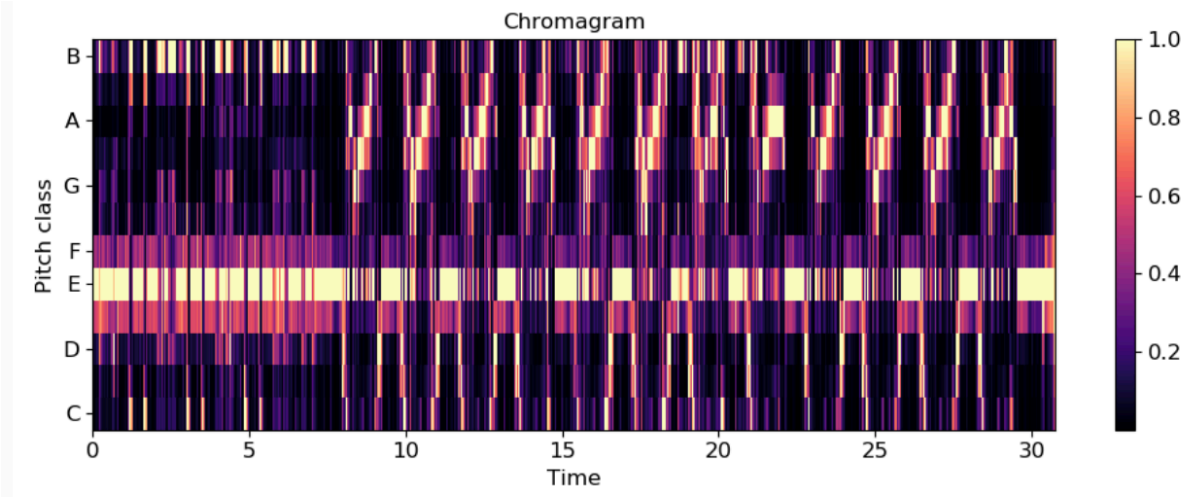

Рисунок 1.6. - Хроматограма

Алгоритм, хроматичного аналізу наступний:

1) Обчислюється хроматична матриця. Використовується швидке перетворення Фур'є для дискримінації спектральної лінії.
2) Зміщення кадру завжди дорівнює чверті довжини кадру швидкого перетворення Фур'є(FFT), таким чином колонки C завжди знаходяться в часових рамках len(FFT)/4/sr, де sr - частота дискретизації.
3) Будується спектрограма з використанням більш короткого вікна

Спектральний контраст.

Даний метод вилучення функцій є досить перспективним, за дослідженнями вчених з університету Цінхуа, спектральний конраст має кращу дискримінацію серед різних типів аудіо ніж MFCC[5].

Спектральний контраст використовується на основі октав. Обчислюється відносний спектральний розподіл, замість середньої огинаючої спектру. Експерименти показали, що функція спектрального контрасту (ФСК) на базі октав добре зарекомендувала себе у класифікація аудіо сигналів.

В загальному, функція спектрального контрасту на основі октав застосовується для представлення відносних спектральних характеристик

звукових сигналів. ФСК на основі октави враховує силу спектральних екстремумів і спектральних долей у кожному піддіапазоні окремо, щоб функція могла представляти відносні спектральні характеристики, апроксимовано відображаючи розподіл гармонічних та негармонічних компонент.

Основною проблемою подібного типу аудіо є те, що у вхідному сигналі досить багато шуму. ФСК може приблизно відображати відносний розподіл гармонічних та негармонічних компонент спектру. Наприклад, MFCC усереднюють спектральний розподіл у кожному піддіапазоні тому втрачають відносну спектральну інформацію. Рахуючи, що два спектри, що мають різний спектральний розподіл, можуть мати схожі спектральні характеристики, середнього спектрального розподілу недостатньо для представлення спектральних характеристик аудіо.

На рисунку 1.7. зображено порівняння ФСК(a) на базі октав та MFCC(b)

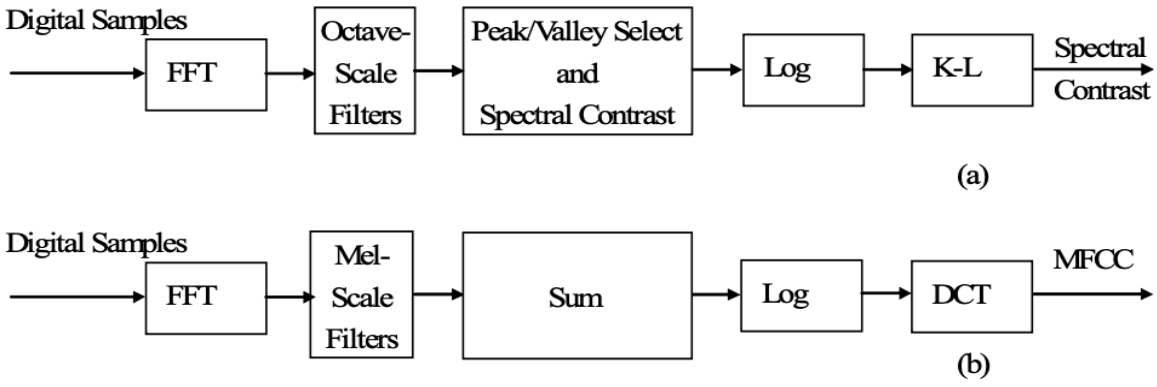

Рисунок 1.7. – ФСК на базі октав та MFCC

На Рис. 1.7.(a) проілюстровано алгоритм оцінки характеристик спектрального контрасту на базі октав. FFT виконується на цифрових вибірках для отримання спектру. Потім частотна область ділиться на піддіапазони деякими фільтрами октавної шкали. Сила спектральних екстремумів, долей їх різниці оцінюється у кожному піддіапазоні. Після перетворення в лог-область необроблений елемент спектрального контрасту відображається в ортогональну проекцію і усереднює

відносність між різними вимірами за допомогою перетворення Кархунена-Лоева.

Теорема Кархунена-Лоева – мінімальне значення норм помилки представлення сигналів на інтервалі протяжністю T досягається при використанні в якості базису власних функцій оператора, ядром якого є кореляційна функція сигналів $R_a(t,\tau)$:

$$\int_{-\frac{T}{2}}^{\frac{T}{2}} R_a(t,\tau)\varphi_k(\tau)d\tau = \lambda_k\varphi_k(t), \qquad (2.23)$$

відповідних N найбільшим власним значенням. Норма помилки:

$$\|\epsilon\|^2_{min} = \|a(t) - \sum_{k=0}^{N-1}\alpha_k\varphi_k(t)\|^2_{min} = \sum_{k=N}^{\infty}\lambda_k. \qquad (2.24)$$

Процедура оцінки MFCC зображена на Рис. 1.7. (b), для порівняння з характеристикою спектрального контрасту на базі октав. Між процедурами (a) та (b) є такі відмінності:

1) Набір фільтрів відрізняється. Спектральний контраст на базі октав, ФСК використовує фільтри октавного масштабу в той час як MFCC використовує мел-масштабні фільтри.

2) Спектральний контраст вилучає силу спектральних екстремумів, долі і їх різницю у кожному піддіапазоні, а MFCC - суми діапазонів амплітуди FFT. Таким чином, функція спектрального контрасту являє спектральні характеристики, а MFCC включає лише усереднену спектральну інформацію, тому спектральний контраст несе більше спектральної інформації, ніж MFCC.

3) На останньому кроці функція спектрального контрасту використовує перетворення Кархунена-Лоева, коли MFCC

використовує перетворення DCT.

Сила спектральних піків та спектральних долей оцінюється як:

$$Peak_k = \log\{\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} x'_{k,i}\}$$
$$Valley_k = \log\{\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} x'_{k,N-i+1}\} \quad (2.25)$$

А їх різниця:

$$SC_k = Peak_k - Valley_k \quad (2.26)$$

, де N – загальне число в *k-му* піддіапазоні.

$$\{SC_k\ ,\ Valley_k\}\ \ (k \in [1,6]) \quad (2.27)$$

, використовується як 12-мірний рядок функції спектрального контрасту.

Хоча спектральний контраст означає різницю в силі між спектральними екстремумами та долями, сила спектральних долей також є.

На рисунку 1.7. зображено фінальний, нормалізований спектральний контраст.

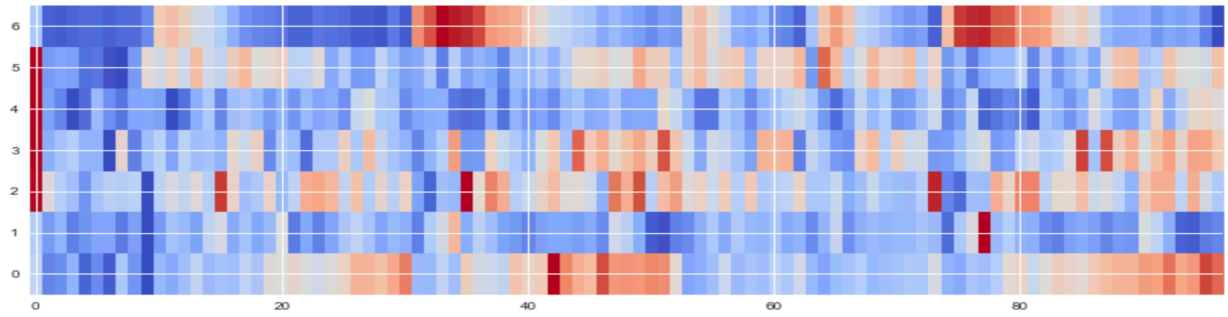

Рисунок 1.7. – Нормалізований спектральний контраст

На рисунку 1.8. зображено спектрограму потужності, на рисунку 1.9. зображено спектральний контраст.

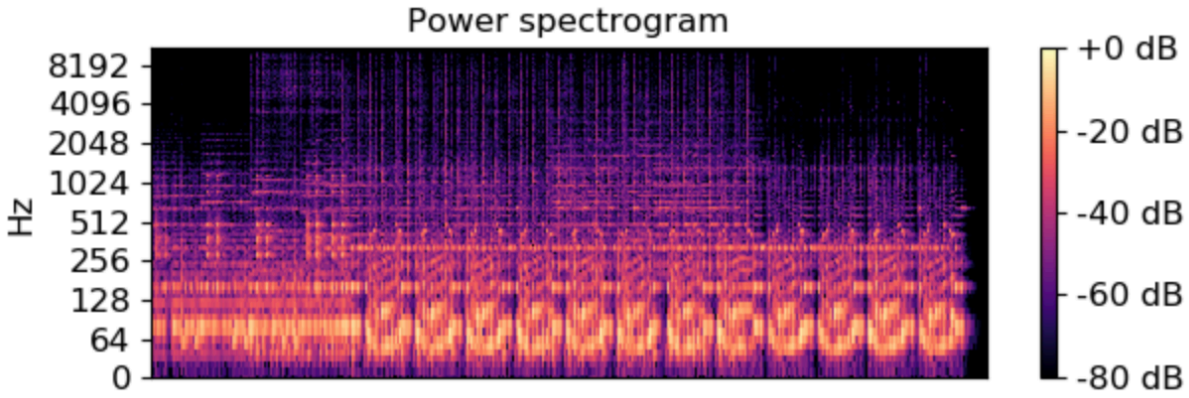

Рисунок 1.8. - Спектрограму потужності

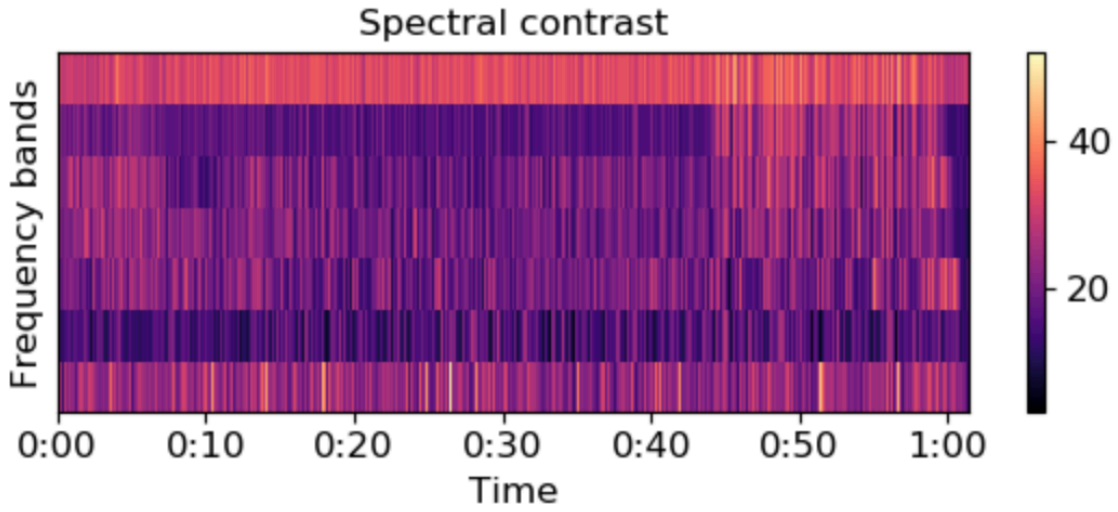

Рисунок 1.9. - Спектральний контраст

Підсумовуючи те, що описано вище, можна зробити висновок, що було вибрано коректні інструменти до вилучення ознак з аудіо, які апроксимують кількість інформації.

3 АРХІТЕКТУРА НЕЙРОННОЇ МЕРЕЖІ ДЛЯ РОБОТИ С ЧАСОВИМИ РЯДАМИ

3.1. Архітектура мереж

У дипломній роботі було досліджено два типи нейронних мереж, кожна з яких має свої особливості, позитивні та негативні сторони.

Так як в роботі було поставлено акцент не на перебір та аналіз різних алгоритмів машинного навчання, що в апріорному уявлення не дадуть результату, а на аналіз та дослідження, побудову такого алгоритму, який спроможний дати максимально можливий результат.

В більшості випадків аудіо- це дискретна чи нескінченна послідовність, сигнали передаються послідовно від початку до кінця,

для роботи з часовими сигналами в середовищі нейронних мереж прийнято використовувати рекурентні нейронні мережі.

RNN – мережа такого типу, що сигнали передаються послідовно від шару до шару (feed-forward neural networks). У рекурентних мережах нейрони обмінюються інформацією між собою, тобто нейрон на часовому проміжку t отримує інформацію як з цього проміжку, так і частину інформації з t-n. Таким чином створюється механізм пам’яті, що дозволяє аналізувати будь-які послідовності де важливий порядок значень. На рисунку 3.1 зображено схему одношарової рекурентної нейронної мережі- на кожному циклі роботи, внутрішній шар отримує інформацію на попередніх кроках.

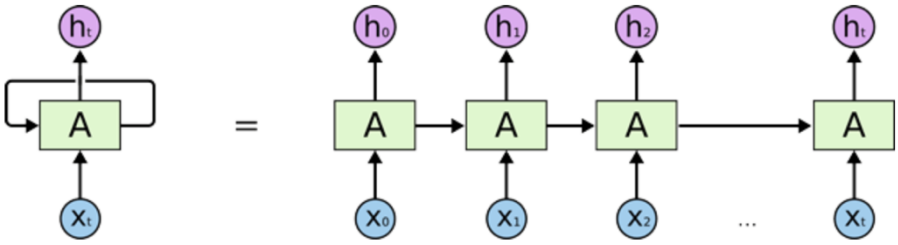

Рисунок 3.1. – Одношарова RNN

Про можливості рекурентних нейронних мереж детально описано в статі Recurrent Neural Networks for Time Series Forecasting [10].

Коли мова йде про рекурентні мережі, мається на увазі, що вони здатні ‘запам’ятовувати’ всю довжину послідовності, але, на жаль, на практиці це не так, звичайна RNN має уявлення тільки про недавні елементи, але досить швидко ігнорує початок часової серії. Це призводить до такого явища як вибух градієнту[11]. Це робить RNN неефективною для тих задач, же потрібно тримати в пам’яті довгі послідовності даних.

Одна із основних ідей RNN полягає в тому, що вони потенційно вміють пов'язувати попередню інформацію з поточним завданням, так, наприклад, знання про попередній кадр відео можуть допомогти в розумінні поточного кадру. Якби RNN володіли такою здатністю, вони були б надзвичайно корисні.

Довга короткострокова пам'ять (Long short-term memory; LSTM) - особливий різновид архітектури рекурентних нейронних мереж, здатна до навчання довготривалим залежностям. Вони були представлені Зеппом Хохрайтером і Юргеном Шмідхубер (Jürgen Schmidhuber) в 1997 році, а потім вдосконалені і популярно викладені в роботах багатьох інших дослідників. Вони прекрасно вирішують цілий ряд різноманітних завдань і в даний час широко використовуються.

LSTM розроблені спеціально, щоб уникнути проблеми довготривалої залежності. Запам'ятовування інформації на довгі періоди часу - це їх звичайна поведінка, а не щось, чого вони насилу намагаються навчитися.

Будь-яка рекурентна нейронна мережа має форму ланцюжка повторюваних модулів нейронної мережі. У звичайній RNN структура одного такого модуля дуже проста, наприклад, він може являти собою один шар з функцією активації tanh (гіперболічний тангенс). На рисунку 3.2 зображено модуль стандартної RNN.

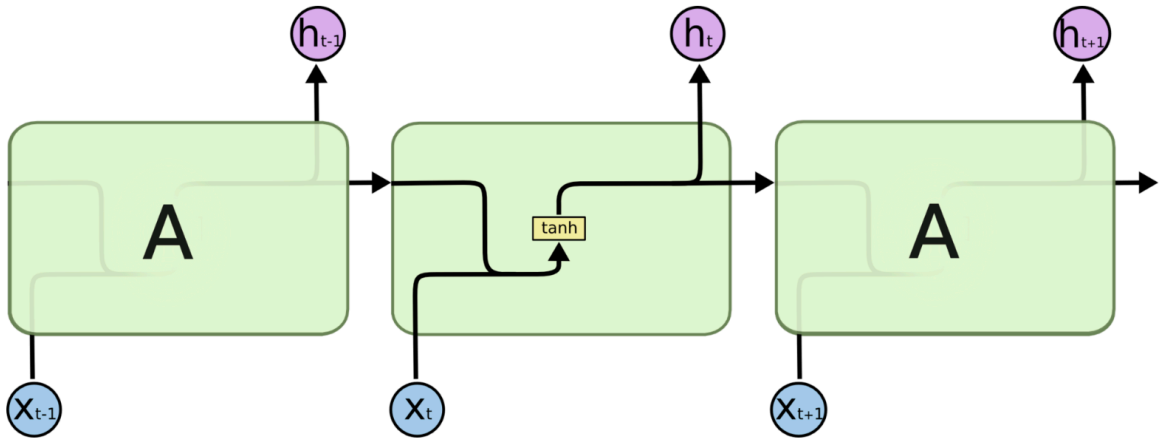

Рисунок 3.2. - Модуль стандартної RNN

Структура LSTM також нагадує ланцюжок, але модулі виглядають інакше. Замість одного шару нейронної мережі вони містять цілих чотири, і ці шари взаємодіють особливим чином. На рисунку 3.3 зображено рекурентний модуль LSTM мережі с чотирьох взаємпов'язаних шарів.

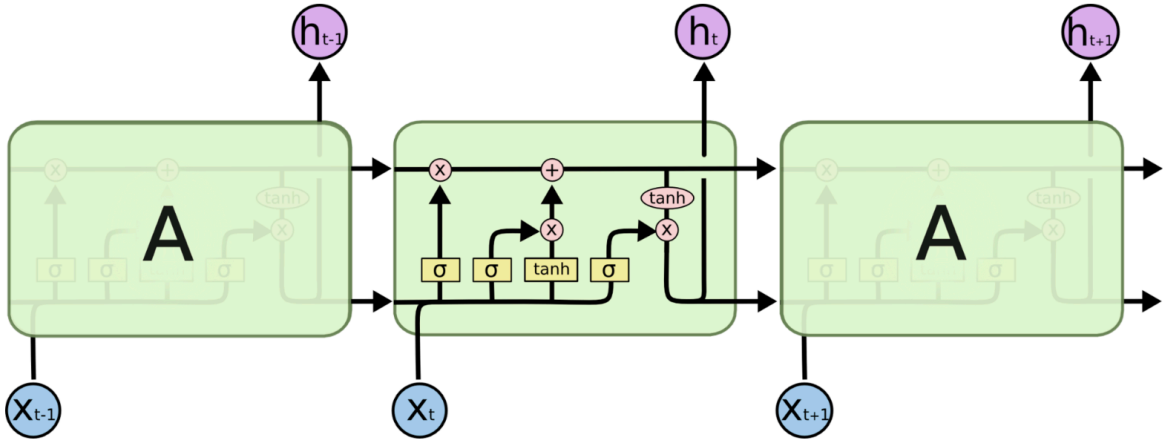

Рисунок 3.3. - Рекурентний модуль

Ключовий компонент LSTM - це стан осередку (cell state) - горизонтальна лінія, що проходить по верхній частині схеми.

Стан осередку нагадує конвеєрну стрічку. Вона проходить безпосередньо через весь ланцюжок, беручи участь лише в декількох лінійних перетвореннях. Інформація може легко текти по ній, не наражаючись зміни.

На рисунку 3.4 зображено внутрішню складову рекурентної комірки.

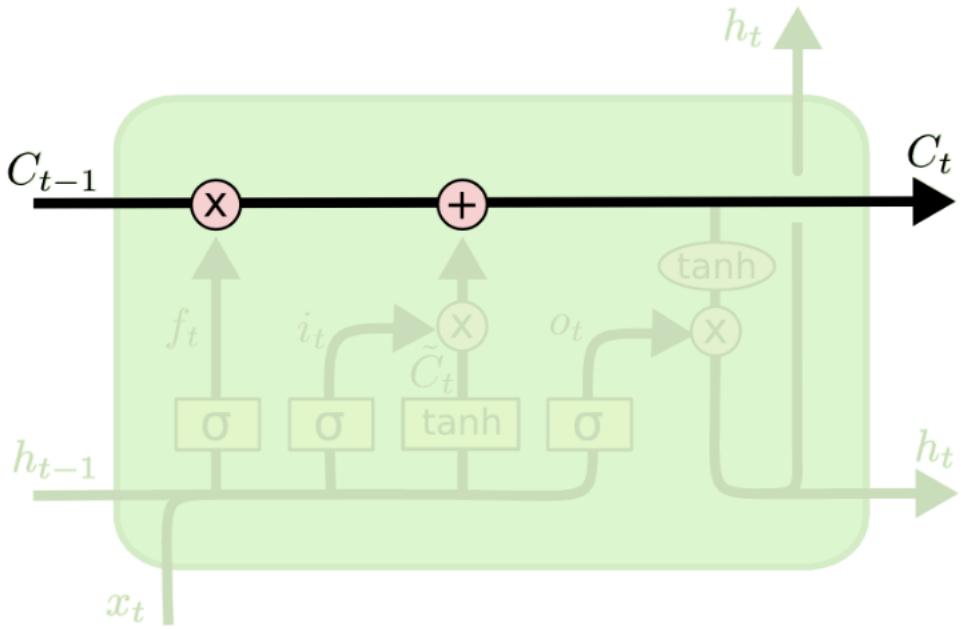

Рисунок 3.4. - Внутрішня складову рекурентної комірки

Проте, LSTM може видаляти інформацію зі стану осередку; цей процес регулюється структурами, званими фільтрами (gates).

Фільтри дозволяють пропускати інформацію на підставі деяких умов. Вони складаються з шару сигмоїдальної нейронної мережі і операції покрапкового множення.

Сигмоїдальний шар повертає числа від нуля до одиниці, які позначають, яку частку кожного блоку інформації слід пропустити далі по мережі. Нуль в даному випадку означає "не пропускати нічого", одиниця - "пропустити все". У LSTM три таких фільтра, що дозволяють захищати і контролювати стан комірки.

Перший крок в LSTM - визначити, яку інформацію можна викинути зі стану осередку. Це рішення приймає сигмоїдальний шар, званий "шаром фільтра забування" (forget gate layer). Він дивиться на $\boldsymbol{h_{t-1}}$та $\boldsymbol{x_t}$ і повертає число від 0 до 1 для кожного числа зі стану комірки. 1 означає "повністю зберегти", а 0 - "повністю викинути".

Повернемося до нашого прикладу - мовної моделі, пророкує наступне слово на підставі всіх попередніх. У цьому випадку стан осередку повинно зберегти іменника, щоб потім використовувати займенники

відповідного роду. Коли ми бачимо нову сутність, ми можемо забути стару.

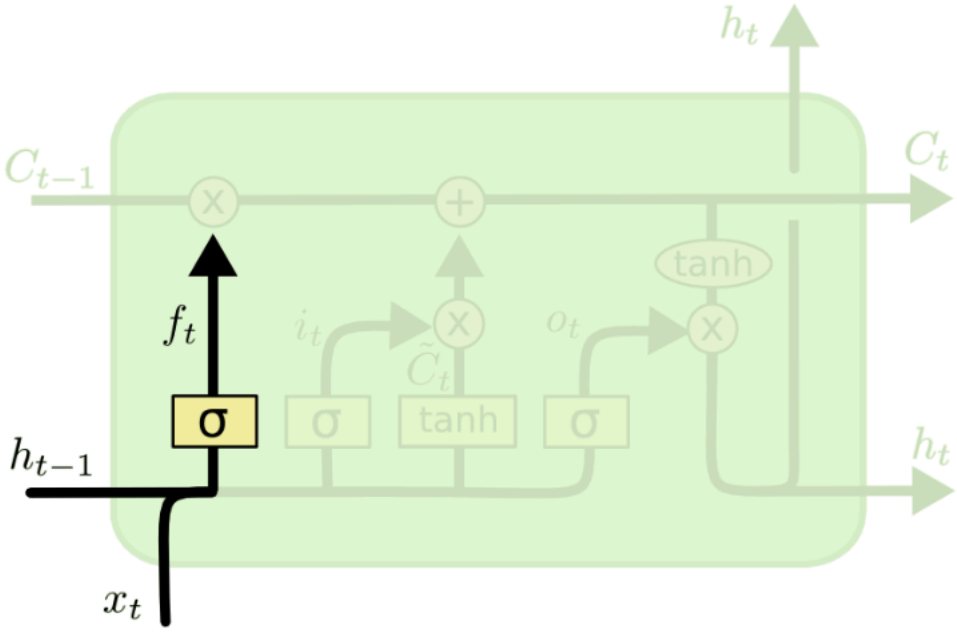

Рисунок 3.5. – Внутрішній стан комірки.

На рисунку 3.5. зображено внутрішній стан комірки, який описано формулою (3.1.)

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right) \tag{3.1.}$$

Наступний крок - вирішити, яка нова інформація буде зберігатися в стані осередки. Цей етап складається з двох частин. Спочатку сигмоїдальний шар під назвою "шар вхідного фільтра" (input layer gate) визначає, які значення слід оновити. Потім tanh-шар будує вектор нових значень-кандидатів $\boldsymbol{C_t}$, які можна додати в стан комірки.

У нашому прикладі з мовної моделлю на цьому кроці ми хочемо додати рід нового іменника, замінивши при цьому старий. Що проілюстровано на рисунку 3.6.

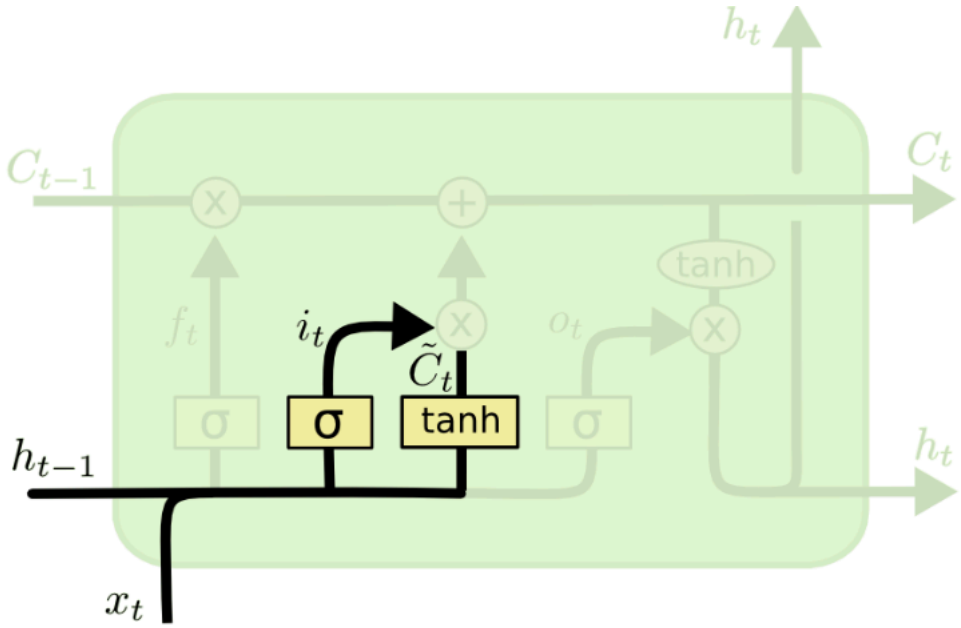

Рисунок 3.6. – Робота комірки

Та описується формулою (3..2)

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C) \tag{3.2}$$

Настав час замінити старе стан комірки $\boldsymbol{C_{t-1}}$ на новий стан $\boldsymbol{C_t}$. Що нам потрібно робити - ми вже вирішили на попередніх кроках, залишається тільки виконати це.

Ми множимо старий стан на $\boldsymbol{f_t}$, забуваючи те, що ми вирішили забути. Потім додаємо $\boldsymbol{i_t * C_t}$. Це нові значення-кандидати, помножені на - на скільки ми хочемо оновити кожне з значень стану.

У разі нашої мовної моделі це той момент, коли ми викидаємо інформацію про рід старого іменника і додаємо нову інформацію.

На рисунку 3.7. зображено стан комірки.

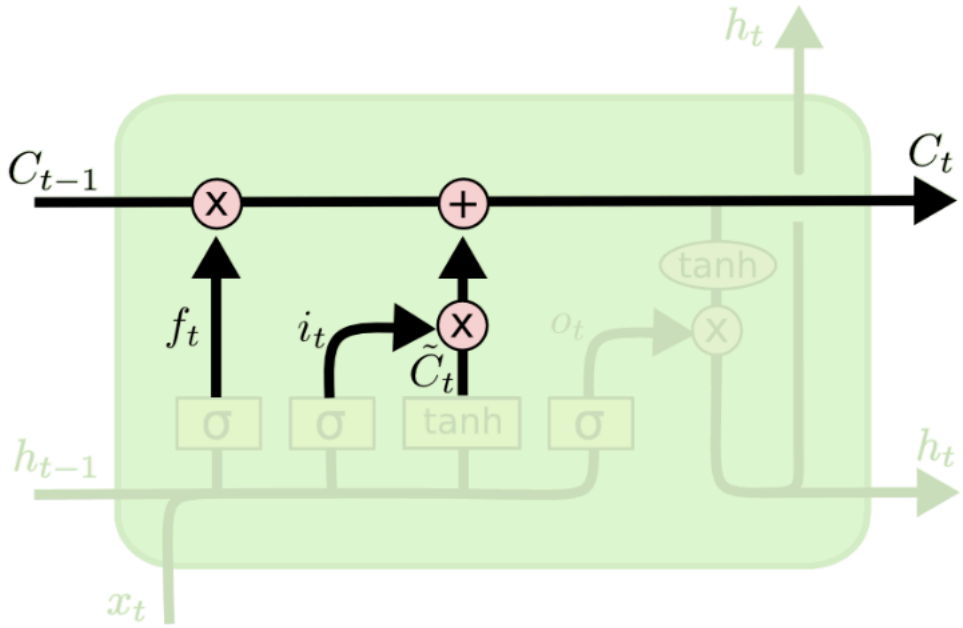

Рисунок 3.7. – Стан комірки

Що описується формулою (3.3)

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.3}$$

Нарешті, потрібно вирішити, яку інформацію ми хочемо отримувати на виході. Вихідні дані будуть засновані на нашому стані осередки, до них будуть застосовані деякі фільтри. Спочатку ми застосовуємо сигмоїдальний шар, який вирішує, яку інформацію зі стану осередку ми будемо виводити. Потім значення стану осередку проходять через tanh-шар, щоб отримати на виході значення з діапазону від -1 до 1, і перемножуються з вихідними значеннями сигмоїдального шару, що дозволяє виводити тільки необхідну інформацію.

На рисунку 3.8. зображено стан комірки

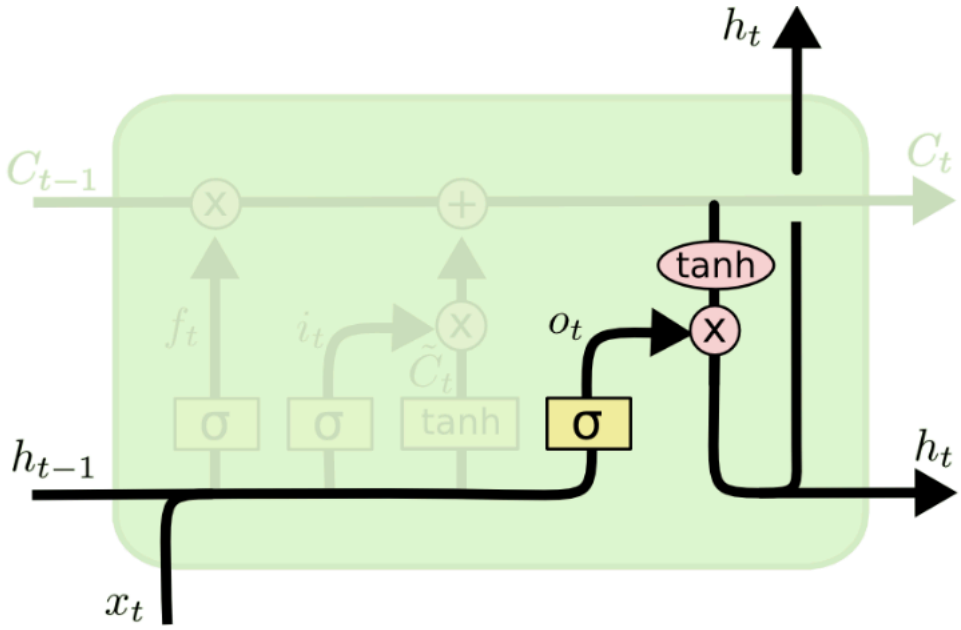

Рисунок 3.8. – Наступний стан комірки

Що описується формулою (3.4).

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right) \tag{3.4}$$

Вище розглянуто звичайну LSTM; але не всі LSTM однакові. Взагалі, здається, що в кожній новій роботі, присвяченій LSTM, використовується своя версія LSTM. Відмінності між ними незначні, але про деякі з них варто згадати.

Одна з популярних варіацій LSTM, запропонована Герсом і Шмідхубер (Gers & Schmidhuber, 2000)[12], характеризується додаванням так званих "оглядових вічок" ("peephole connections"). З їх допомогою шари фільтрів можуть бачити стан комірки, що описується формулою (3.5).

$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t]\ +\ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t]\ +\ b_i\right) \tag{3.5}$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t]\ +\ b_o\right)$$

Інші модифікації включають об'єднані фільтри "забування" і вхідні фільтри. У цьому випадку рішення, яку інформацію слід забути, а яку запам'ятати, приймаються не окремо, а спільно. Ми забуваємо будь-яку інформацію тільки тоді, коли необхідно записати щось на її місце. Ми додаємо нову інформацію з стан комірки тільки тоді, коли забуваємо стару. Це проілюстровано на рисунку 3.9

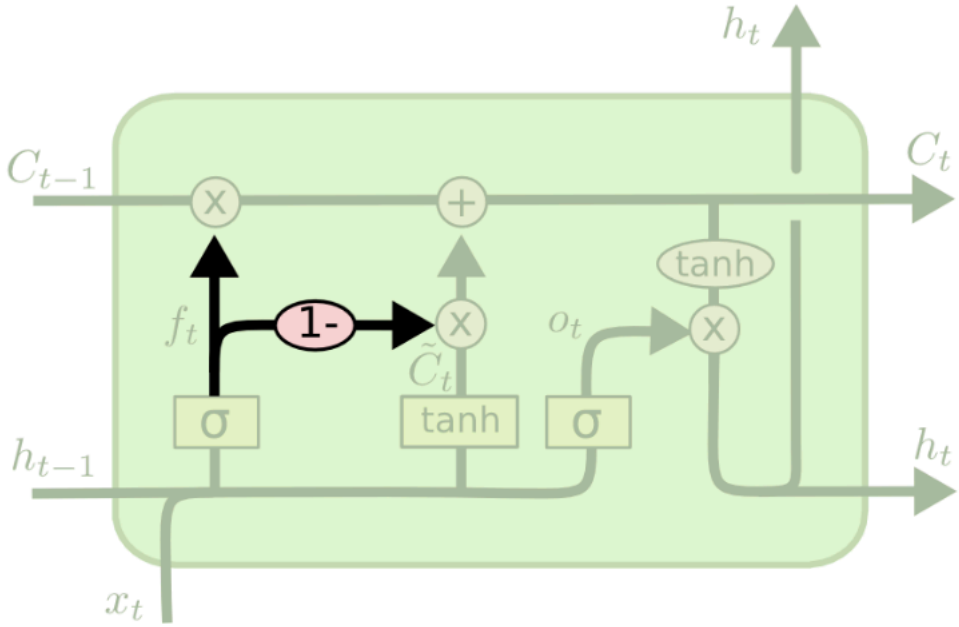

Рисунок 3.9. – Передача інформації в комірці

Рисунок описується формулою (3.6.)

$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t \qquad (3.6.)$$

Трохи більше відрізняються від стандартних LSTM керовані рекурентні нейрони (Gated recurrent units, GRU), вперше описані в роботі Cho, et al (2012)[13] У ній фільтри «забування» і входу об'єднують в один фільтр «оновлення» (update gate). Крім того, стан комірки об'єднується з прихованим станом, є й інші невеликі зміни. Побудована в результаті модель простіше, ніж стандартна LSTM, і популярність її неухильно зростає. Це проілюстровано на рисунку 3.10 та описується формулою (3.7).

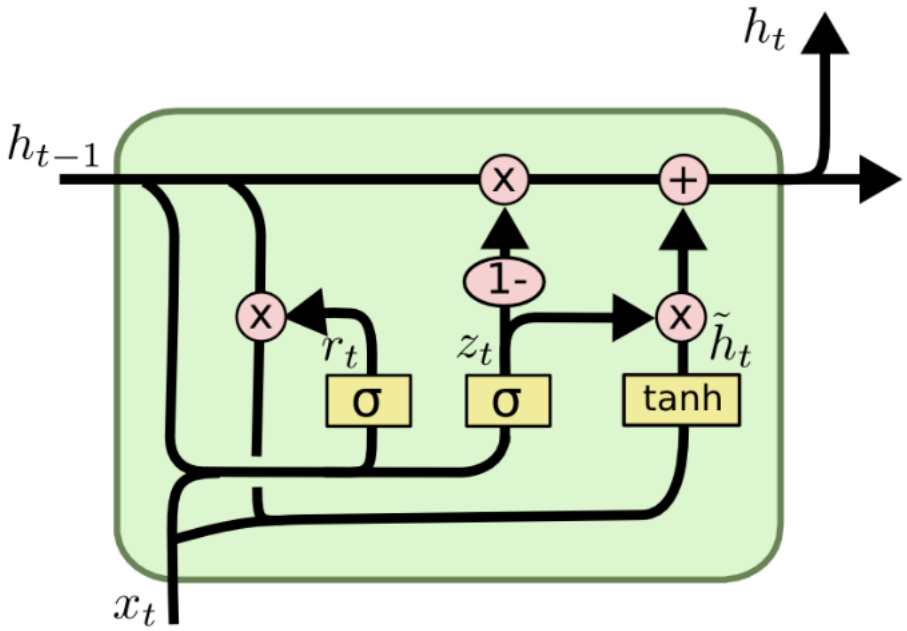

Рисунок 3.10. – Прихований стан комірки

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \quad (3.7.)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Вище розглянуто лише кілька найбільш примітивних варіацій LSTM. Існує безліч інших модифікацій, як, наприклад, глибокі керовані рекурентні нейронні мережі (Depth Gated RNNs), представлені в роботі Yao, et al (2015) [14]. Є й інші способи вирішення проблеми довгострокових залежностей, наприклад, Clockwork RNN Яна Кутніка (Koutnik, et al., 2014 року) [15].

Крім RNN було використано та досліджено Dilated causal convolutions(DCC).

DCC – авторегресійна статистична модель нейронної мережі. Загальна ймовірність сигналу $\boldsymbol{x} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_T}\}$, розкладається на добуток ймовірностей наступним чином:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}) \tag{3.8}$$

Отже, кожен зразок аудіо $\boldsymbol{x_t}$ обумовлений зразками на всіх попередніх часових кроках. Подібно до PixelCNNs (van den Oord et al., 2016a; b)[16], розподіл умовної ймовірності моделюється стеком згорткових шарів. У мережі немає шарів об'єднання і вихід моделі має ту ж часову розмірність, що і вхідний. Модель виходить категоричного розподілу над наступним значенням $\boldsymbol{x_t}$ з шаром softmax і його оптимізують для максимізації логарифмічної достовірності. Оскільки логарифмічність ймовірності є вразливою, ми налаштовуємо гіперпараметри на набір перевірок і можемо легко виміряти, що модель перенавчається, або недостатньо навчилась.

На рисунку 3.11. проілюстровано стек причинних згорткових шарів.

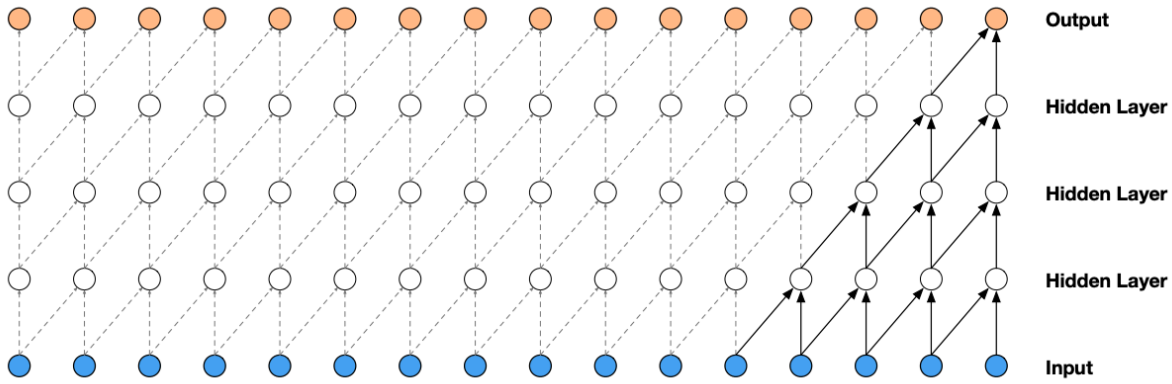

Рисунок 3.11. - Стек причинних згорткових шарів

Основним компонентом моделі є причинні згортки. Використовуючи причинні згортки, ми стаємо впевнені, що модель не може порушити порядок, у якому ми моделюємо дані: прогноз $\boldsymbol{p(x_{t+1} \mid x_1, \ldots, x_t)}$, що прогнозується моделлю в тимчасовому кроці t, не може залежати від будь-якого майбутнього часу $\boldsymbol{x_{t+1}}$, $\boldsymbol{x_{t+2}}$…,$\boldsymbol{x_T}$, як показано на рисунку 3.1.11. Для зображень еквівалент причинної згортки є маскувальна згортка (van den

Oord et al., 2016a)[17], яка може бути реалізована шляхом побудови маски

і виконує елементне множення цієї маски з ядром згортки перед її застосуванням. Для 1-D даних, таких як аудіо, можна легше реалізувати це шляхом перенесення виходу.

Під час навчання умовні прогнози для всіх тимчасових кроків можуть бути зроблені паралельно, тому що всі тимчасові кроки наземної істини x відомі. При генерації з використанням моделі прогнози є послідовними: після прогнозування кожної вибірки вона повертається в мережу для прогнозування наступної вибірки.

Оскільки моделі з причинними згортками не мають повторення сполучень, вони звичайно є швидше для тренування, ніж RNN, що особливо помітно для дуже довгих послідовностей. Одна з проблем причинно-послідовної згортки в тому, що вони вимагають великого набору даних для збільшення рівня натренованості мережі. Як, наприклад, на рисунку. 3.1.11. рецептивне поле складає всього 5. Для збільшення рецептивного поля було прийнято рішення збільшувати згортки.

Розширена згортка (також звана хрестовиною або згорткою з отворами) - це згортка, в якій фільтр застосовується до області, що перевищує його довжину, пропускаючи вхідні значення з певним кроком, це еквівалентно згортці з великим фільтром, отриманим з вихідного фільтра шляхом розширення його нулями, але значно ефективніше. Розширена згортка ефективно дозволяє мережі працювати на більший масштаб, ніж. Це схоже на об'єднання або згортку згорток, але тут результат має той же розмір, що вхід. На рисунку (3.12) зображені причинні згортки для розширень 1, 2, 4 та 8. Розгорнуті згортки раніше використовувалися в різних контекстах, наприклад, обробка сигналу (Holschneider et al., 1989; Dutilleux, 1989)[18] і сегментація зображення (Chen et al., 2015; Yu & Колтун, 2016)[19].

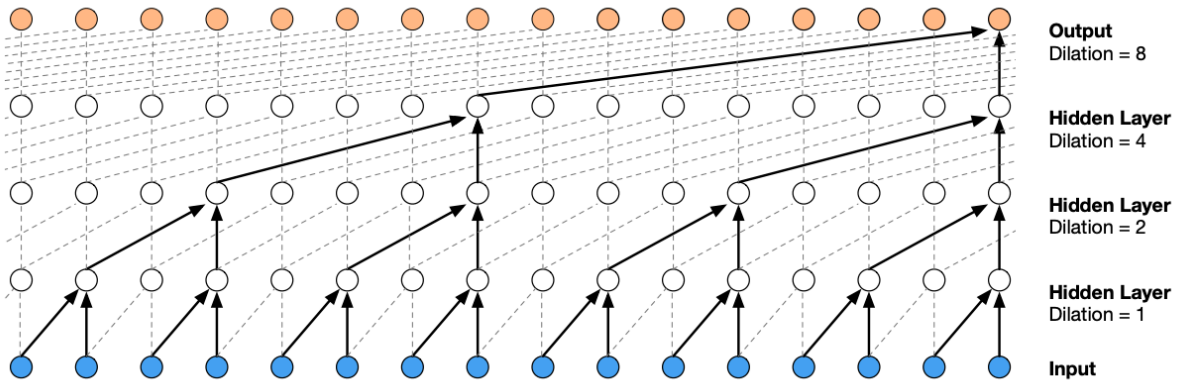

Рисунок 3.12. - Причинні згортки для розширень 1, 2, 4 та 8

Складні розширені згортки дозволяють мережі мати дуже великі рецептивні поля з декількома шарами, зберігаючи при цьому вхідне розширення по всій мережі, а також ефективність обчислень.

У мережі розширення подвоюється для кожного шару до межі, а потім повторюється. Сенс цієї конфігурації двоякий. По-перше, експоненціальне збільшення коефіцієнту розширення призводить до експоненціального зростання рецептивного поля з глибиною (Yu & Koltun, 2016)[20]. Наприклад, кожен 1, 2, 4..,512 блок має рецептивне поле розміром 1024 та може розглядатись як більш ефективний і розпізнавальний (нелінійний) аналог згортки 1 × 1024. По-друге, складаючи ці блоки - збільшується об'єм моделі і розмір рецептивного поля.

Один з підходів до моделювання умовних розподілів $\boldsymbol{p(x_t \mid x_1, \ldots, x_{t-1})}$ з індивіду аудіо- використовується змішана модель, така що мережа з змішаною щільністю (Bishop, 1994)[20]. Тим не менш, Ван ден Оорд і ін. (2016a)[21] показали, що розподіл softmax має тенденцію працювати краще, навіть коли дані неявно безперервні (як у випадку інтенсивності пікселів зображення або значень аудіо). Один через те, що категоріальний розподіл є більш гнучким і може легше моделювати довільний розподіл, тому що він не робить ніяких припущень про їх розмірність. Так як необроблений звук зазвичай зберігається в

послідовності 16-бітних цілих чисел (по одному на часовий крок), шар softmax повинен вивести 65536 імовірностей з часовим шагом для моделювання всіх можливих знаків. Щоб зробити це більш зручним, спочатку застосовується компактне перетворення $\boldsymbol{\mu}$-закону (ITU-T, 1988) для даних, а потім квантувати його до 256 можливих значень:

$$f(x_t) = \operatorname{sign}(x_t)\frac{\ln(1+\mu|x_t|)}{\ln(1+\mu)}, \qquad (3.9.)$$

, де $-\mathbf{1} < \boldsymbol{x_t} < \mathbf{1}$ та $\boldsymbol{\mu}$=255. Це нелінійне квантування виробляє значно краще реконструює, ніж проста схема лінійного квантування. Спеціально для звукового сигналу було виявлено, що відновлений сигнал після квантування звучав дуже схоже на оригінал.

Вже було згадано кілька різних способів збільшити розмір рецептивного поля моделі, а саме: збільшення кількості стадій розширення, з використанням більшої кількості шарів, великих фільтрів, більших коефіцієнтів розширення, або їх поєднання. Додатковий підхід полягає у використанні окремого, меншого контекстного стеку що обробляє довгу частину звукового сигналу і локально обумовлює більшу модель, яку обробляє тільки менша частина звукового сигналу (обрізана в кінці). Можна використовувати декілька стеків контексту з різною довжиною та кількістю прихованих одиниць. Стек з більшими рецептивними полями має менше одиниць на шар. Стек контексту також може мати шари об'єднання, які виконуються з меншою частотою. Це зберігає обчислювальні вимоги на розумному рівні.

Вище було проаналізовано глибоку генеративну модель аудіо даних, яка працює безпосередньо з рівень сигналу. Модель є авторегресивною і об'єднує причинно-наслідкові фільтри з розширеними згортками, щоб дозволити їх рецептивним полям зростати експоненційно з глибиною, що

є важливою для моделювання далеких часових залежностей в аудіосигналах. Було показано, як модель може бути обумовлена на інших входах у глобальному (наприклад, ідентифікація мови) або локальному (наприклад, мовні особливості) сенсі.

Підсумовуючи, можна зробити висновок, що DCC є потужним та перспективним вибором серед архітектур нейронних мереж, яка забезпечує високу швидкість та якість роботи.

3.2. Структура алгоритмів

Основною ціллю даної дипломної роботи було дослідження та порівняння алгоритмів нейронних мереж, а також дослідження функцій, які вилучаються з аудіо сигналів, для передачі на алгоритм.

Програму розділено на дві частини, кожна з яких виконую свою функцію та містить певну кількість класів.

Перша частина- завантаження, конвертація, обробка, архівування та вилучення функцій з аудіо. Всі ці речі рознесено по функціям класу SoundFeatureData. Клас містить такі функції, як:

1) __init__ - ініціалізує загальний масив значень та параметр hop_length, який задає кількість вибірок між послідовними кадрами.
2) load_preprocess_data – функція, що завантажує дані з директорії приводить до одного формату та ділить аудіо на тестувальну, тренувальну, та валідаційну вибірку. А також архівує аудіо та зберігає їх в локальну директорію, тип .npy.
3) load_deserialize_data – функція, яка дозволяє завантажити архівовані дані.
4) extract_audio_features – функція, що дозволяє вилучити корисні функції з аудіо сигналу та сформувати вибірку для передачі на алгоритм машинного навчання.

5) one_hot – допоміжна функція, яка дозволяє кодувати класи аудіо сигналів у бінарний вигляд, для передачі на нейронну мережу.

6) path_to_audiofiles – допоміжна функція, допомагає без проблем завантажити дані.

На рисунку 3.13 зображено скріншот коду, що описує ініціалізацію класу.

```
import numpy as np
import librosa
import math
import re
import os

class SoundFeatureData:

    'Music audio features for sound classification'
    hop_length = None
    genre_list = ['air_conditioner', 'car_horn', 'children_playing', 'dog_bark', 'drilling', 'engine_idling', 'gun_shot',
'jackhammer','siren','street_music']

    dir_trainfolder = "./data/train"
    dir_devfolder = "./data/val"
    dir_testfolder = "./data/test"
    dir_all_files = "./data"

    train_X_preprocessed_data = 'data_train_input.npy'
    train_Y_preprocessed_data = 'data_train_target.npy'
    dev_X_preprocessed_data = 'data_validation_input.npy'
    dev_Y_preprocessed_data = 'data_validation_target.npy'
    test_X_preprocessed_data = 'data_test_input.npy'
    test_Y_preprocessed_data = 'data_test_target.npy'

    train_X = train_Y = None
    dev_X = dev_Y = None
    test_X = test_Y = None

    def __init__(self):
        self.hop_length = 512
        self.timeseries_length_list = []
```

Рисунок 3.13. – Ініціалізація класу SoundFeatureData

Друга частина – безпосередньо завантаження оброблених та готових до подачі на алгоритм матриць та тренування нейронної мережі. Так як тренування нейронної мережі займає значних ресурсів комп'ютера та часу, було вирішено максимально оптимізовано побудувати структуру класів та функцій.

Частина містить такі класи, як:

1) DatasetLoad – клас, який дозволяє організувати подачу даних на алгоритм невеликими партіями, задля того, щоб забезпечити розпаралелювання та задіяти всі логічні процесори комп'ютера. При ініціалізації класу задаються такі параметри, як: розмір партії, флаг перемішування, кількість потоків, флаг використання графічного

процесору. На рисунку (3.14) зображено скріншот даного класу, його функції та ініціалізацію.

```
class DatasetLoad(Dataset):

    def __init__(self, data):
        self.data = data

    def __len__(self):
        return len(self.data[0])

    def __getitem__(self, index):

        feature = self.data[0][index]
        target = self.data[1][index]

        return feature, target

train_dataset = DatasetLoad(TRAIN)
test_dataset = DatasetLoad(TEST)

device = 'cuda'
train_loader = torch.utils.data.DataLoader(train_dataset,
                                           batch_size = batch_size, shuffle = True,
                                           num_workers=4,pin_memory="cuda" in device)
test_loader = torch.utils.data.DataLoader(test_dataset,
                                          batch_size = batch_size, shuffle = True,
                                          num_workers=4, pin_memory="cuda" in device)

```

Рисунок 3.14. - Ініціалізація класу DatasetLoad

2) RNN – клас, який описує рекурентну нейронну мережу та її складові.

На рисунку (3.15) зображено скріншот даного класу, його функції та ініціалізацію.

```python
class RNN(nn.Module):
    def __init__(self):
        super(RNN, self).__init__()

        self.rnn = nn.LSTM(
            input_size=33,
            hidden_size=128,
            num_layers=1,
            batch_first=True,
           # dropout = 0.35
        )
        self.dropout = nn.Dropout(0.35)
        self.fc = nn.Linear(128, 32)
        self.BN = nn.BatchNorm1d(128)

        self.out = nn.Linear(32, 10)

    def forward(self, x):
        r_out, (h_n, h_c) = self.rnn(x, None)
        x = self.fc(r_out)
        x = self.BN(x)
        x = self.dropout(x)
        x = F.relu(x)
        out = self.out(x[:, -1, :])
        return out

rnn = RNN()
rnn.to(device)
print(rnn)
```

Рисунок 3.15. - Ініціалізація класу RNN

Далі класи використовуються у функціях train та val, що характеризують процес тренування та валідації нейронної мережі. Тренування було організовано таким чином, що в алгоритм мережі поступово передавались нові партії даних, та в режимі реального часу проходили валідацію. На рисунку 3.16. зображено скріншот функції тренування.

```python
def train(epoch):
    epoch_loss, epoch_loss2 = 0, 0
    for step, (b_x, b_y ) in enumerate(train_loader):
        optimizer.zero_grad()
        output = rnn(b_x.cuda())
        loss = loss_func(output, torch.argmax(b_y.cuda(), 1))
        epoch_loss += loss.item()
        loss.backward()
        optimizer.step()

    print("===> Epoch {} Complete: Avg. Loss: {:.7f}".format(epoch, epoch_loss / len(train_loader)))
```

Рисунок 3.16. – Функція тренування

3) CausalConv1d – клас, що описує згорткову авторегресійну 1D мережу, та її складові. На рисунку 3.17. зображено скріншот даного класу, його функції та ініціалізацію.

```python
class CausalConv1d(nn.Module):

    def __init__(self, kernel_size, in_channels, out_channels, dilation):
        super(CausalConv1d, self).__init__(self)

        self.kernel_size = kernel_size
        self.in_channels = in_channels
        self.dilation = dilation

        self.conv1d = torch.nn.Conv1d(in_channels, out_channels,
                                      kernel_size, stride=1,
                                      padding=(kernel_size-1),
                                      dilation=dilation)

    def forward(self, seq):
        seq_ = seq.permute(1,2,0)
        conv1d_out = self.conv1d(seq_).permute(2,0,1)
        return conv1d_out[0:-(self.kernel_size-1)]
```

Рисунок 3.17. – Ініціалізація класу CausalConv1d

Працюючи над архітектурою нейронної мережі, для досягнення максимального результату, розробнику потрібно слідкувати за такими речами, як метрика точності, метрика функції втрат та баланс цих метрик між валідаційною та тренувальною вибіркою. Є кілька варіантів залежностей; overfitting, underfitting та normalfitting.

Overfitting означає, що модель занадто ускладена для набору даних і занадто добре навчається на тренувальних даних, але погано узагальнює, словом перенавчається.

Underfitting означає, що модель є занадто простою і її складності не вистачає для навчання.

Normalfitting означає, що модель добре збалансована та навчання йде добре.

На рисунку 3.18, відповідно зображені три випадки:

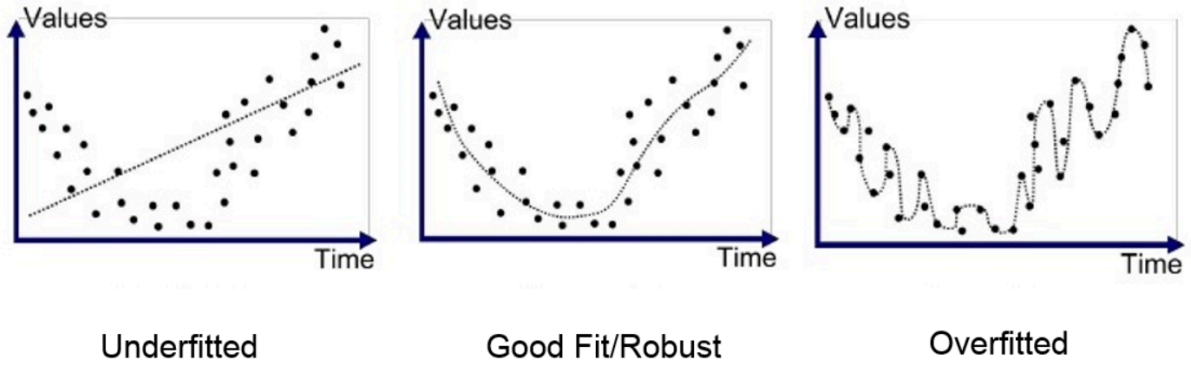

Рисунок 3.18.- Випадки залежності функції втрат

Мережі RNN за своєю природою здатні до перенавчання, тому розробнику потрібно контролювати та підбирати таку архітектуру моделі, щоб забезпечити стійкість. Було протестовано моделі з різною кількістю рекурентних шарів, емпіричним шляхом було вирішено застосувати два рекурентні шари, та набір лінійних шарів в комбінації з батч нормалізацією.

Батч нормалізація – метод, що дозволяє підвищити продуктивність моделі та стабілізувати її роботу. Суть полягає в тому, щоб нормалізувати дисперсію та математичне очікування після кожного шару. Нормалізували вхідні дані нейронних мереж до нульового середнього і постійного стандартного відхилення.

Відомо протягом десятиліть, щоб бути корисними для навчання нейронних мереж. З виникненням глибоких мереж, Пакетна нормалізація (BN) природно розширює цю ідею через проміжні шари в глибині мережі, хоча з причин швидкості нормалізація виконується через міні-партії, а не весь набір тренувань. Сьогодні в машинному навчальному співтоваристві мало що виникає BN прискорює навчання, підвищує швидкість навчання і підвищує точність узагальнення і BN успішно застосовували у всіх областях глибокого навчання. Однак, незважаючи на його незаперечний успіх, є ще незначний консенсус щодо того, чому переваги BN такі виражений. У своїй оригінальній публікації [23] Іоффе та Сегеді припускають, що БН може полегшити «Внутрішній коваріантний зсув» -

тенденція розподілу активацій до дрейфу під час навчання, таким чином впливаючи на входи в наступні шари. Однак інші пояснення, такі як поліпшення стабільності були запропоновані одночасні оновлення або кондиціонування. Натхненний останніми емпіричними знаннями про глибоке навчання , у цій роботі ми прагнемо роз'яснити ці невизначені інтуїції, поклавши їх на тверду експериментальну основу. Покажемо, що активації і градієнти в глибоких нейронних мережах без BN мають тенденцію бути важкими. Зокрема, під час на початку набору дивергенції, невелика підмножина активацій (зазвичай в глибокому шарі) «вибухає». Типова практика уникнення такої дивергенції полягає в тому, щоб встановити швидкість навчання на достатньо малі, щоб це було відсутність крутого напрямку градієнта може призвести до розбіжності. Проте, невеликі показники навчання не дають значного прогресу вздовж плоских напрямків оптимізації ландшафту і може бути більш схильним до зближення з різким локальних мінімумів з можливим погіршенням генералізації. BN уникає вибуху активації, багаторазово виправляючи всі активації як нульові і одиничні стандартне відхилення. Завдяки цій «безпеці» можна навчати мережі з великим рівнем навчання 32-а Конференція з систем обробки нейронної інформації, Монреаль, Канада. ставки, оскільки активації не можуть зростати вперед, оскільки їхні засоби та відхилення нормалізуються. SGD з великими швидкостями навчання дає більш швидку конвергенцію вздовж плоских напрямків оптимізації ландшафт і менш ймовірно застрягти в різких мінімумах.

Досліджуємо інтервал життєздатних темпів навчання для мереж з і без BN і укладаємо що BN набагато більше прощає дуже великі темпи навчання. Експериментально ми демонструємо, що активації в глибоких мережах без BN різко зростають з глибиною, якщо швидкість навчання занадто велика. Нарешті, ми досліджуємо вплив випадкової ініціалізації ваги на градієнти в мережі і встановити зв'язки з останніми результатами теорії випадкових матриць, які припускають, що традиційні схеми

ініціалізації можуть бути не дуже придатними для мереж з багатьма шарами - якщо тільки BN не використовується підвищити стійкість мережі до поганих умов.

Як і в [23], ми перш за все розглянемо BN для згорткових нейронних мереж. Як вхідний, так і вихідний шар BN - це чотиривимірні тензори, які ми називаємо Ib, c, x, y і Ob, c, x, y, відповідно. Розміри, що відповідають прикладам в межах партії b, каналу c і двох просторових розмірів x, y відповідно. Для вхідних зображень канали відповідають каналам RGB. BN застосовується однаково нормалізація для всіх активацій в даному каналію

$$O_{b,c,x,y} \leftarrow \gamma_c \frac{I_{b,c,x,y} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c \qquad \forall\, b, c, x, y. \tag{3.10}$$

Останнім шаром рекурентної моделі є лінійне стиснення нейронів до потрібної кількості класів та функція softmax.

в математиці функція softmax, також відома як softargmax або нормалізована експоненціальна функція, є функцією, яка приймає вхідний вектор K дійсних чисел, і нормалізує її у розподіл ймовірностей, що складається з K ймовірностей. Тобто, до застосування softmax, деякі векторні компоненти можуть бути негативними або більшими, ніж один; і може не дорівнювати 1; але після застосування softmax, кожен компонент буде знаходитися в інтервалі {displaystyle (0,1)} (0,1), і компоненти будуть додаватися до 1, так що вони можуть бути інтерпретовані як ймовірності. Більш того, більші вхідні компоненти будуть відповідати більшим вірогідність. Softmax часто використовується в нейронних мережах, щоб відобразити ненормалізований вихід мережі до розподілу ймовірностей над прогнозованими виходами.

Функція softmax часто використовується в кінцевому шарі класифікатора на основі нейронної мережі. Такі мережі зазвичай навчаються в режимі логарифмічних втрат (або крос-ентропії), даючи нелінійний варіант поліноміальної логістичної регресії.

Авторегресійна CNN для отримання хорошої якості роботи, потребує великого рецептивного поля, тому було прийнято рішення побудувати восьмишарову модель нейронної мережі, яка включає послідовне енкодування вектору функцій, та їх декодування- для вилучення ознак, що більше впливають на кінцевий результат.

Важливим є те, що після кожного шару використовуються функції активації нейронів.

У штучних нейронних мережах функція активації вузла визначає вихід цього вузла з урахуванням входу або набору входів. Стандартну комп'ютерну мікросхему можна розглядати як цифрову мережу функцій активації, яка може бути "ON" (1) або "OFF" (0), залежно від входу. Це схоже з поведінкою лінійного персептрона в нейронних мережах. Однак тільки нелінійні функції активації дозволяють таким мережам обчислювати нетривіальні задачі, використовуючи лише невелике число вузлів. У штучних нейронних мережах цю функцію також називають передатною функцією.

Деякі бажані властивості в функції активації включають: Нелінійна - коли функція активації є нелінійною, то двошарова нейронна мережа може бути доведена як універсальний апроксиматор функції. Функція активації ідентичності не задовольняє цієї властивості. Коли кілька шарів використовують функцію активації ідентичності, вся мережа еквівалентна одношаровій моделі. Діапазон - Коли діапазон функції активації є кінцевим, методи навчання на основі градієнта мають тенденцію бути більш стабільними, оскільки презентації шаблонів значно впливають лише на обмежені ваги. Коли діапазон нескінченний, тренування, як правило, є більш ефективним, оскільки презентації шаблонів значно впливають на більшість ваг. В останньому випадку, як правило, потрібні менші показники навчання. Безперервно диференційований - Це властивість бажано (RELU не є постійно диференційованим і має деякі проблеми з оптимізацією на основі градієнта, але це все ще можливо) для включення

методів оптимізації на основі градієнта. Функція активації двійкового кроку не диференціюється при 0, і вона розрізняє 0 для всіх інших значень, тому методи градієнтної обробки не можуть зробити жодного прогресу. Монотонний - Коли функція активації монотонна, поверхня помилки, пов'язана з одношаровою моделлю, гарантовано є опуклою. Гладкі функції з монотонною похідною - в деяких випадках вони краще узагальнюються. Апроксимує ідентичність поблизу походження - коли функції активації володіють цією властивістю, нейронна мережа буде ефективно вивчати, коли її ваги ініціалізуються малими випадковими значеннями. Коли активаційна функція не наближається до ідентичності походження, особливу обережність необхідно використовувати при ініціалізації ваг.

4 АНАЛІЗ РЕЗУЛЬТАТІВ НАВЧАННЯ ТА РОБОТИ НЕЙРОННОЇ МЕРЕЖІ

4.1. Аналіз роботи мереж.

Було натреновано та зроблено детальний аналіз двох видів нейронних мереж, а саме рекурентну нейронну мережу та згорткову.

В цьому розділі наведено графіки та аналіз роботи RNN, CNN та їх порівняння.

Основною числовою оцінкою якості роботи моделі є функція втрат.

Нейронні мережі навчаються з використанням стохастичного градієнтного спуску і вимагають вибору функції втрати при проектуванні та налаштуванні моделі. Існує багато функцій втрат. Нейронні мережі навчаються з використанням процесу оптимізації, який вимагає функції втрат для обчислення помилки моделі. Максимальна ймовірність забезпечує основу для вибору функції втрати при навчанні нейронних мереж і моделей машинного навчання в цілому. Крос-ентропія і середня квадратична помилка є двома основними типами функцій втрати, які використовуються при навчанні нейронних мережевих моделей. "Градієнт" у градієнтному спуску відноситься до градієнта помилок. Модель з заданим набором ваг використовується для прогнозування і обчислення похибки прогнозів. Алгоритм градієнтного спуску прагне змінити ваги так, щоб наступна оцінка знижувала похибку, тобто алгоритм оптимізації здійснював навігацію по градієнту (або нахилу) помилки.

У контексті алгоритму оптимізаційна функція, яка використовується для оцінки рішення кандидата (тобто набір ваг), називається цільовою функцією. Ми можемо прагнути максимізувати або мінімізувати цільову функцію, це означає, що ми шукаємо рішення, яке має найвищий або найнижчий бал відповідно. Як правило, в нейронних мережах ми прагнемо мінімізувати помилку. Таким чином, цільова функція часто називається

функцією витрат або функцією втрат, а значення, обчислене функцією втрат, називається просто "втратою". Функцію, що мінімізується або максимізується, називають цільовою функцією або критерієм. Коли вона зводиться до мінімуму, ми можемо називати це функцією витрат, функцією втрати або функцією помилки. Функція втрат відіграє важливу роль в тому, що вона повинна точно відганяти всі аспекти моделі в один номер таким чином, щоб поліпшення цього числа було ознакою кращої моделі. Функція втрат знижує всі різні хороші та погані аспекти можливо складної системи до єдиного числа, скалярного значення, що дозволяє оцінювати та порівнювати рішення кандидатів.

При обчисленні похибки моделі в процесі оптимізації необхідно вибрати функцію втрат. Це може бути складною проблемою, оскільки функція повинна враховувати властивості проблеми та бути мотивованою проблемами, які є важливими для роботи та зацікавленими сторонами. Тому важливо, щоб функція вірно відображала цілі.

Існує багато функцій, які можна використовувати для оцінки похибки набору ваг у нейронній мережі. Ми віддаємо перевагу функції, де простір рішень-кандидатів відображається на плавному ландшафті, який алгоритм оптимізації може розумно переміщатися за допомогою ітеративних оновлень ваг моделі. Оцінка максимальної правдоподібності, або MLE, є основою для виведення кращих статистичних оцінок параметрів з історичних даних навчання: саме те, що ми намагаємося зробити з нейронною мережею. Максимальна ймовірність прагне знайти оптимальні значення параметрів шляхом максимізації функції правдоподібності, отриманої з навчальних даних. У нас є навчальний набір з однією або декількома вхідними змінними, і ми вимагаємо, щоб модель оцінювала параметри ваги моделі, які найкраще відображають приклади входів у вихідну або цільову змінну. Враховуючи введення, модель намагається зробити прогнози, які відповідають розподілу даних цільової змінної. Під максимальною ймовірністю функція втрат оцінює, наскільки тісно

розподіл прогнозів, зроблених моделлю, відповідає розподілу цільових змінних у навчальних даних. Втрата крос-ентропії часто просто називають «крос-ентропія», «логарифмічна втрата», «втрата логістики» або «втрата журналу». Кожна передбачена ймовірність порівнюється з фактичним вихідним значенням класу (0 або 1), і розраховується оцінка, яка карає ймовірність на основі відстані від очікуваного значення. Штраф - логарифмічний, пропонуючи невеликий бал за малі відмінності (0,1 або 0,2) і величезний бал за велику різницю (0,9 або 1,0). Втрата крос-ентропії мінімізується, коли менші значення представляють кращу модель, ніж великі значення. Модель, яка прогнозує досконалі ймовірності, має крос-ентропію або втрату логарифма 0,0. Крос-ентропія для задачі прогнозування двійкового або двох класів фактично обчислюється як середня перехресна ентропія всіх прикладів. Крос-ентропія може бути розрахована для класифікації декількох класів. Класи були кодовані одним способом, що означає, що для кожного значення класу існує двійкова функція, і передбачення має передбачати ймовірності для кожного з класів. Потім перехресну ентропію підсумовують по кожній двійковій функції і усереднюють по всіх прикладах у наборі даних. Враховуючи рамки максимальної правдоподібності, ми знаємо, що ми хочемо використовувати крос-ентропію або середню квадратичну функцію втрат помилок при стохастичному спуску градієнта. Проте ми можемо або не хочемо повідомляти про роботу моделі, використовуючи функцію втрат. Наприклад, логарифмічна втрата є складною для інтерпретації, особливо для зацікавлених сторін практикуючих фахівців, що не мають машинного навчання. Те ж саме можна сказати і про середню квадратичну помилку. Замість цього, може бути більш важливим повідомляти про точність і середньоквадратичну похибку для моделей, що використовуються для класифікації і регресії відповідно. Також може бути бажано вибрати моделі на основі цих показників замість втрат. Це є важливим фактором, оскільки модель з мінімальними втратами може не бути моделлю з кращою

метрикою, яка є важливою для зацікавлених сторін проекту. Хорошим підрозділом для розгляду є використання втрат для оцінки та діагностування того, як добре навчається модель. Це включає в себе всі міркування процесу оптимізації, такі як перенавчання, недостатнє оснащення та конвергенція. Потім може бути обраний альтернативний показник, який має значення для зацікавлених сторін проекту, щоб оцінити продуктивність моделі та виконати вибір моделі. Використовується для оцінки та діагностування оптимізації моделі. Використовується для оцінки та вибору моделей у контексті роботи. Таку саму метрику можна використовувати для обох проблем, але більш ймовірно, що проблеми процесу оптимізації будуть відрізнятися від цілей роботи, і будуть потрібні різні оцінки. Тим не менш, часто буває, що поліпшення втрат покращується або, в гіршому випадку, не впливає на показник.

В якості функції втрат було обрано CrossEntropyLoss, на рисунку 4.1. зображено графік loss функції від 1 до 100 епох.

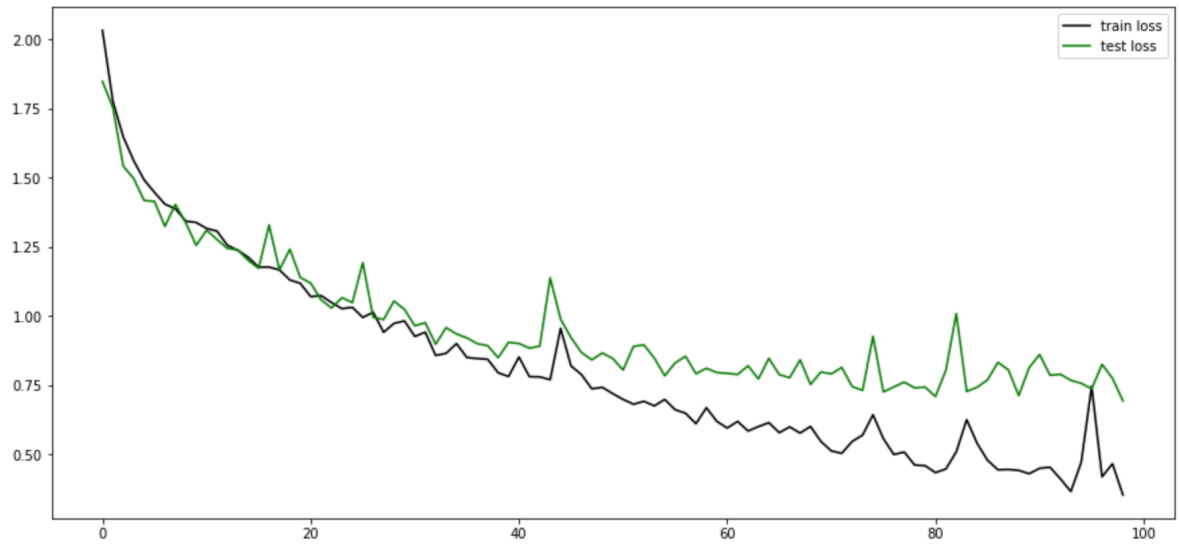

Рисунок 4.1. – Графік функції втрат

Loss функція є характеристикою навчання мережі, але це не завжди означає, що модель добре класифікує, або виконує поставлену задачу.

Оцінка алгоритму машинного навчання є важливою частиною будь-

якого проекту. Модель може дати вам задовольняючі результати, якщо оцінюватиметься за допомогою метрики: tell_city_score, але може дати погані результати при оцінці щодо інших показників, таких як logarithmic_loss або будь-який інший такий показник. У більшості випадків ми використовуємо точність класифікації для вимірювання продуктивності нашої моделі, однак недостатньо, щоб насправді судити про нашу модель. На цій посаді ми розглянемо різні типи доступних оціночних показників.

Класифікаційна точність - це те, що ми звичайно маємо на увазі, коли ми використовуємо термін точність. Це відношення числа правильних прогнозів до загальної кількості вхідних вибірок. Вона добре працює, тільки якщо є однакова кількість зразків, що належать кожному класу. Наприклад, вважаємо, що в нашому навчальному наборі є 98% зразків класу A і 2% зразків класу B. Тоді наша модель може легко отримати точність навчання на 98%, просто передбачивши кожну навчальну вибірку, що належить до класу A. Коли одна й та ж модель тестується на тестовому наборі з 60% зразками класу A і 40% зразків класу B, то точність випробування знизиться до 60%. Класифікаційна точність велика, але дає нам помилкове відчуття досягнення високої точності. Реальна проблема виникає, коли вартість неправильної класифікації вибірок малого класу дуже висока. Якщо ми маємо справу з рідкісним, але смертельним захворюванням, вартість невдачі діагностування хвороби хворої людини набагато вища, ніж вартість відправлення здорової людини на додаткові тести.

Плутанина матриця, як випливає з назви, дає нам матрицю як вихідний і описує повну продуктивність моделі. Припустимо, що ми маємо задачу двійкової класифікації. Ми маємо деякі зразки, що належать до двох класів: ТАК або НІ. Також у нас є власний класифікатор, який передбачає клас для заданого вхідного зразка. При тестуванні нашої моделі на 165 зразків ми отримуємо наступний результат, рисунок 4.2.

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Рисунок 4.2. – Приклад відповідей бінарної класифікації

Є 4 важливі терміни:

- True Positives: випадки, в яких ми передбачали ТАК, а фактичний випуск також був ТАК.
- True Negatives: випадки, коли ми передбачали НІ, а фактичний випуск - НІ.
- False Positives: випадки, в яких ми передбачали YES, а фактичний результат - NO.
- False Negatives: випадки, коли ми передбачали НІ, а фактичний результат - ТАК.

Площа під кривою (AUC) - одна з найбільш широко використовуваних показників для оцінки. Він використовується для задачі двійкової класифікації. AUC класифікатора дорівнює ймовірності того, що класифікатор ранжує випадково вибраний позитивний приклад вище, ніж випадково вибраний негативний приклад. Перш ніж визначити AUC, давайте зрозуміємо два основні терміни:

- True Positive Rate (Sensitivity) : True Positive Rate визначається як TP / (FN + TP). Справжня позитивна ставка відповідає частці позитивних даних, які правильно вважаються позитивними, щодо всіх позитивних даних.
- False Positive Rate (Specificity) : False Positive Rate визначається як FP / (FP + TN). Помилкові позитивні показники відповідають частці негативних даних, які помилково вважаються позитивними, по відношенню до всіх негативних точок даних.

Помилкові позитивні значення і істинний позитивний коефіцієнт мають значення в діапазоні [0, 1]. FPR та TPR обчислюються при порогових значеннях, таких як (0.00, 0.02, 0.04,…., 1.00) і малюється графік. AUC - площа під кривою ділянки False Positive Rate проти істинної позитивної швидкості в різних точках [0, 1]. На рисунку 4.3 зображено приклад ROC AUC.

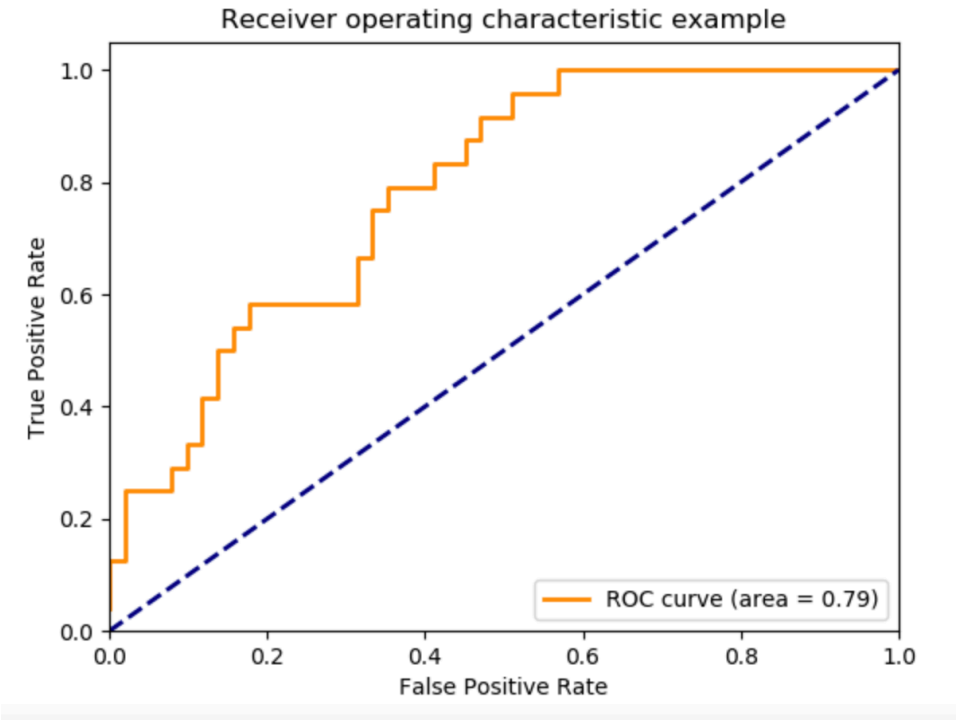

Рисунок 4.3. – Приклад метрики ROC AUC

Очевидно, AUC має діапазон [0, 1]. Чим більше значення, тим краще продуктивність моделі.

Для оцінки якості класифікації, було використано такі метрики як: accuracy,precision, recall, f1-score. А також матрицю плутанини.

На рисунку 4.4. зображено графік тестувальної та тренувальної вибірки на метриці accuracy, як видно з графіку, точність тестувальної вибірки дещо нижче, ніж точність тренувальної, це говорить про те, що архітектура моделі була підібрана вдало.

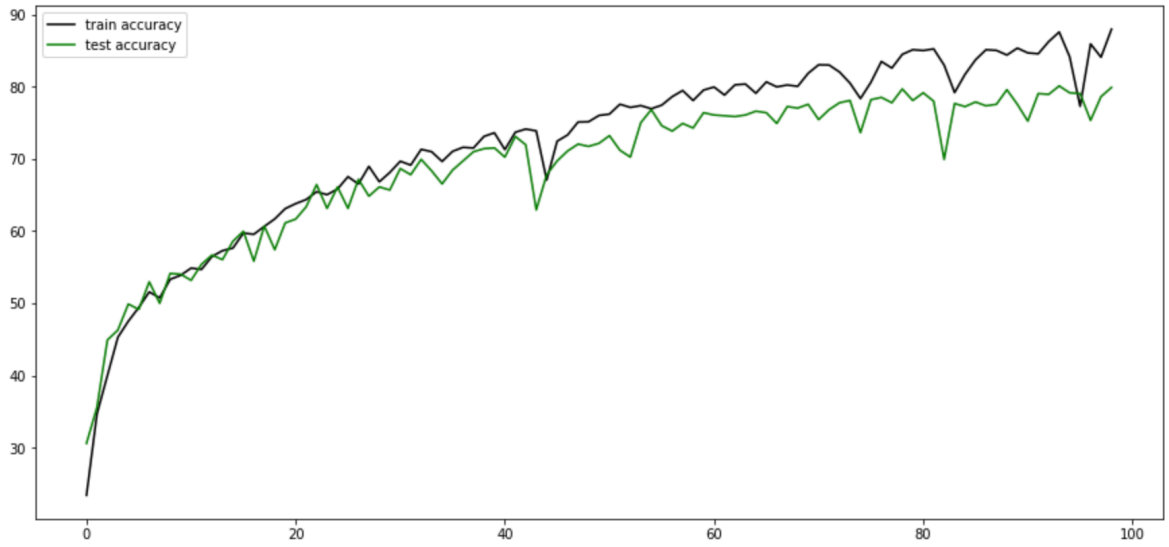

Рисунок 4.4. - Графік тестувальної та тренувальної вибірки на метриці accuracy

Для повного аналізу та розуміння роботи моделі, на рисунку 4.5. зображено таблицю відгуку про класифікацію, а саме ймовірність належності кожного класу, а також зважені та усередненні результати.

	precision	recall	f1-score	support
class 0	0.86	0.84	0.85	105
class 1	0.82	0.61	0.70	44
class 2	0.71	0.75	0.73	106
class 3	0.86	0.76	0.80	90
class 4	0.78	0.81	0.79	114
class 5	0.93	0.88	0.90	115
class 6	0.65	0.92	0.76	36
class 7	0.79	0.75	0.77	114
class 8	0.90	0.95	0.92	112
class 9	0.71	0.75	0.73	108
micro avg	0.81	0.81	0.81	944
macro avg	0.80	0.80	0.80	944
weighted avg	0.81	0.81	0.81	944

Рисунок 4.5. – Відгук про класифікацію RNN

Також, для достовірності вищенаведеної інформації, використаємо матрицю плутанини, як фінальний показник точності класифікації на

тестувальній вибірці, слід зауважити, що матриця є нормалізованою, та числа подані у відсотках, рисунок 4.6.

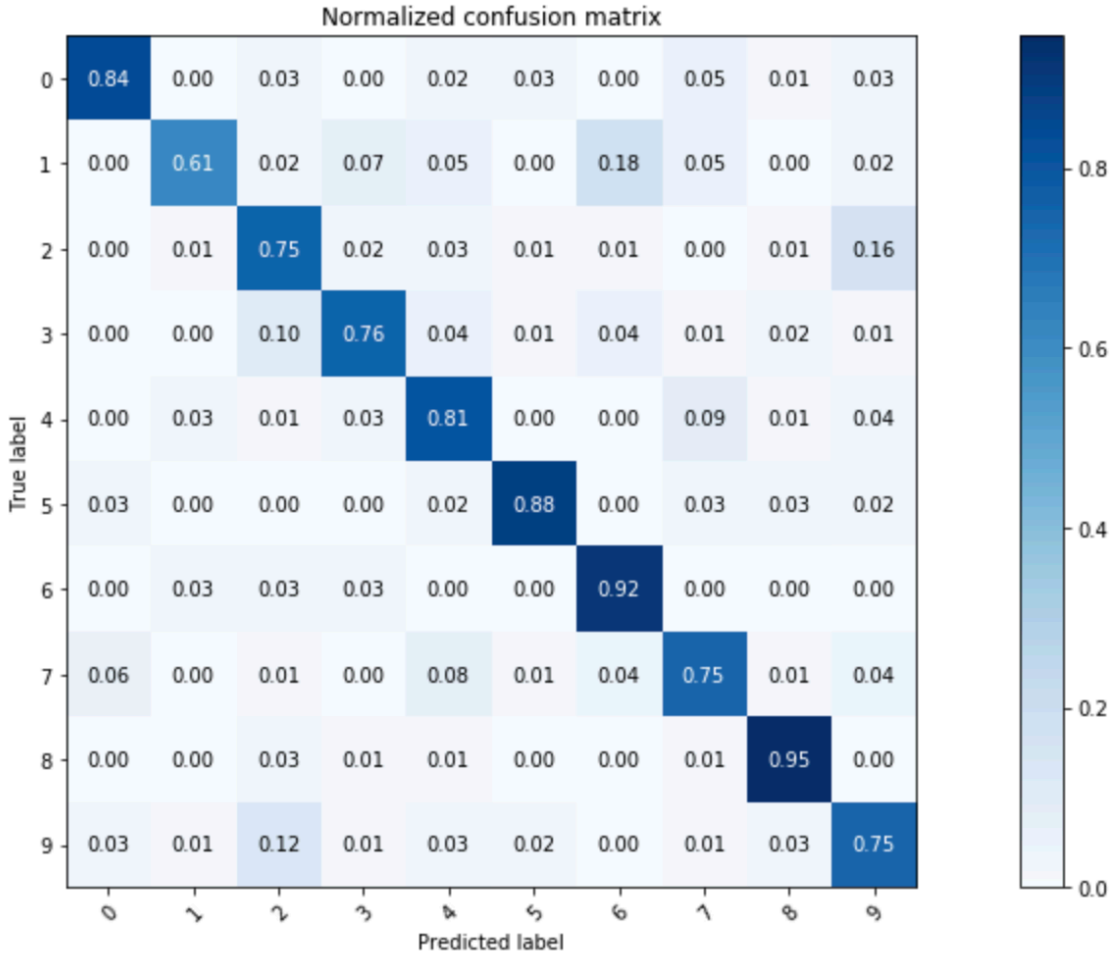

Рисунок 4.6. – Матриця плутанини RNN

Підсумовуючи результати RNN слід сказати, що було отримано досить непогані результати як на тренувальній так і на тестувальній вибірці, модель цілком впоралась з поставленою задачею і відпрацювала так, як було заплановано.

Щодо згорткової нейронної мережі, порівнюючи за аналогічними метриками та функцією втрат, можна побачити, що градієнт loss функції згорткових мереж не таких стабільний та плавний, як в рекурентних, це можна побачити на графіку рисунку 4.7, але тим не менш, найважливішим є те, що градієнт сходиться до нуля.

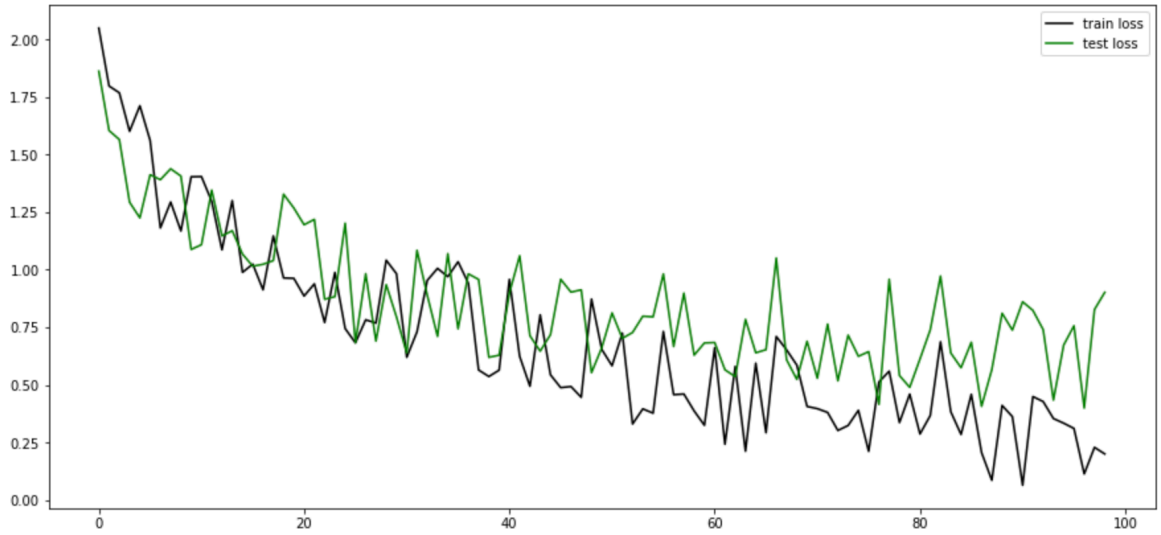

Рисунок 4.7. – Функція втрат CNN

Про результат метрики accuracy слід сказати те, що згорткова модель після дев'яностої епохи почала злегка перенавчатись, саме тому по графіку(рисунок 4.8.) можна побачити, що точність на тренувальній вибірці досягає високого результату, тим не менше, на тестувальній результат теж допустимо високий, тому таке перенавчання допустиме в рамках даної моделі.

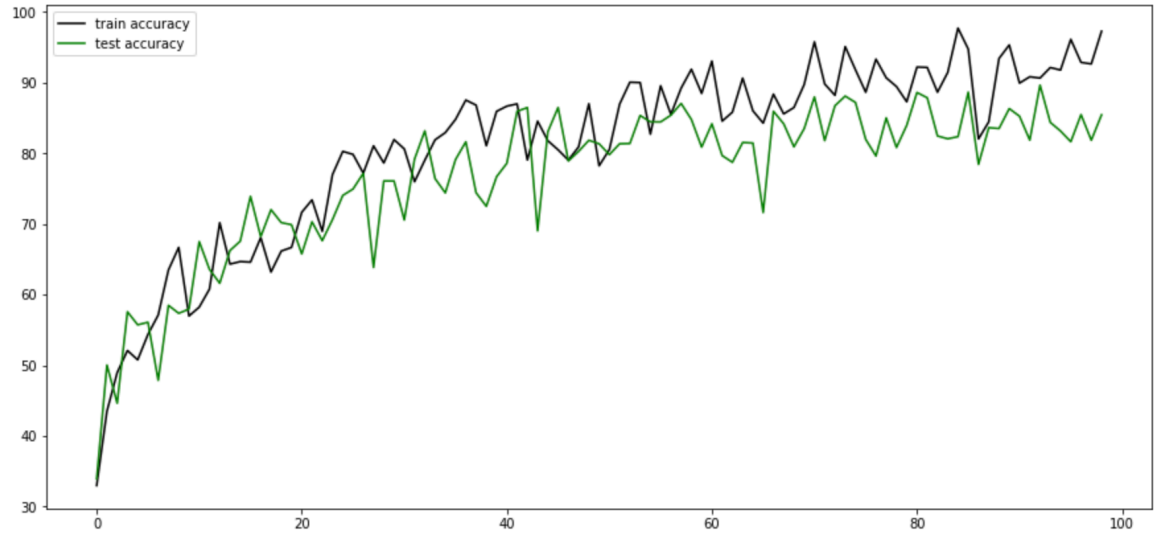

Рисунок 4.8. – Точність RNN

Як було описано вище, важливими критеріями оцінки якості роботи моделі є відгук класифікації по precision, recall, f1-score, а також матриці

плутанини, результати зображено на рисунках 4.9 та 4.10 відповідно.

	precision	recall	f1-score	support
class 0	0.82	0.88	0.85	104
class 1	0.89	0.74	0.81	57
class 2	0.71	0.84	0.77	112
class 3	0.86	0.85	0.86	95
class 4	0.81	0.85	0.83	117
class 5	0.89	0.92	0.90	122
class 6	0.89	0.92	0.90	36
class 7	0.79	0.80	0.79	109
class 8	0.88	0.80	0.84	102
class 9	0.87	0.70	0.77	106
micro avg	0.83	0.83	0.83	960
macro avg	0.84	0.83	0.83	960
weighted avg	0.83	0.83	0.83	960

Рисунок 4.9. – Відгук про класифікацію CNN

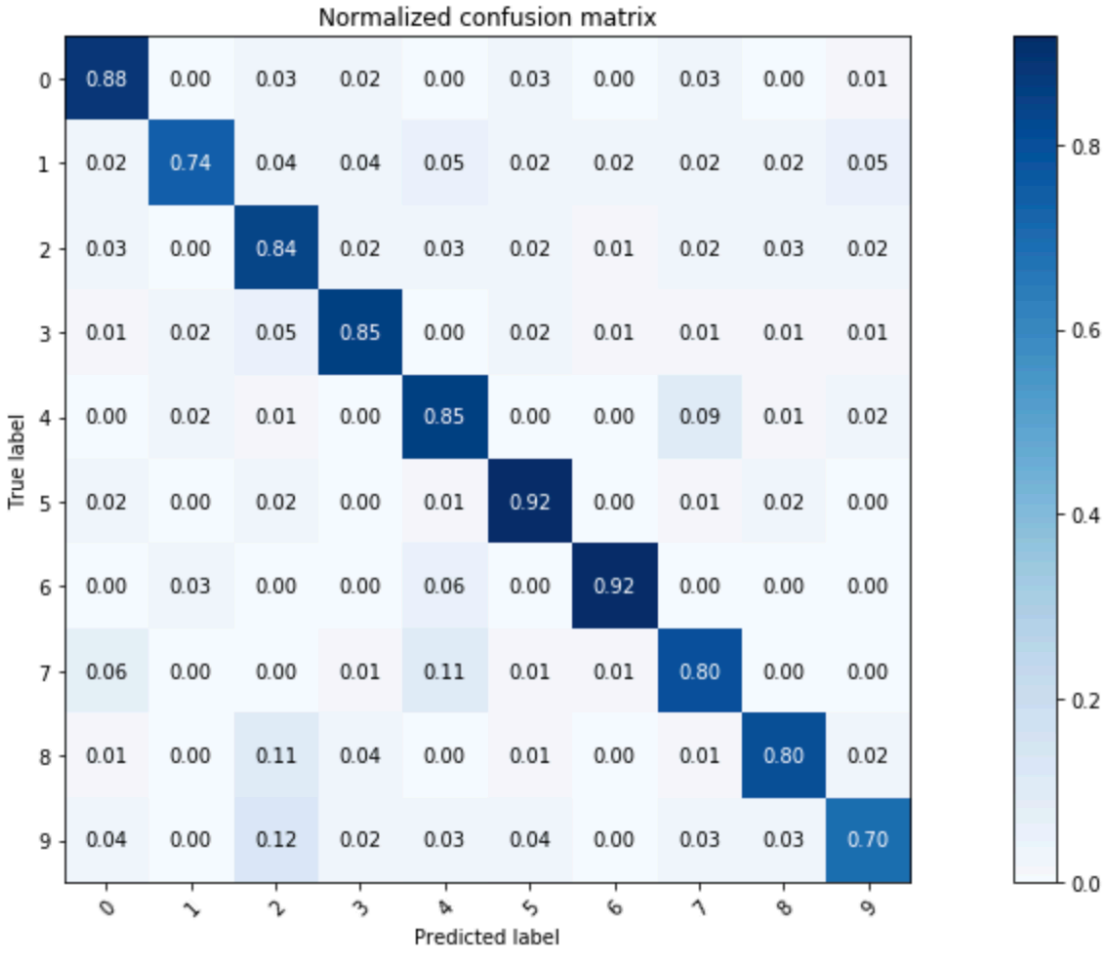

Рисунок 4.10. – Матриця плутанини CNN

Порівнюючи два види архітектур нейронних мереж, на рівні з функцією втрат та метриками точності, важливою характеристикою є час тренування мережі. Враховуючи той фактор, що для різних архітектур потрібна різна кількість епох для навчання, та різний час подачі даних на модель, було прийнято рішення використовувати час за який модель проходить одну епоху, це і буде характеристикою швидкості роботи мережі. На рисунку 4.11. зображено залежність від часу кожної зі ста епох, для згорткової та рекурентної мережі.

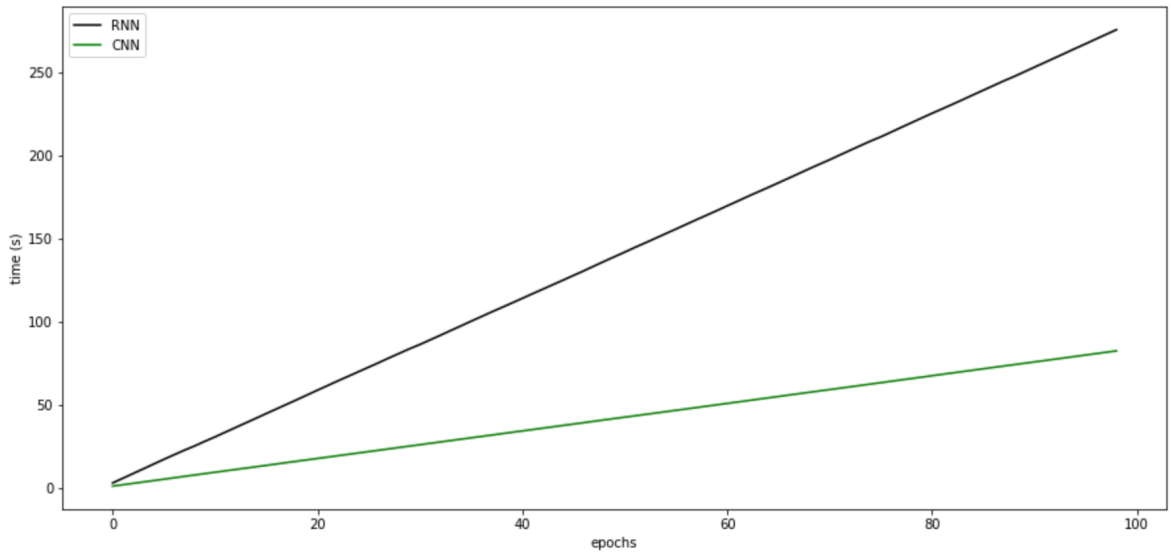

Рисунок 4.11. – Час тренування мереж

Як видно з графіку часу, згорткова нейронна мережа практично вдвічі швидша ніж рекурентна, це пояснюється тим, що в згортковій мережі відбувається менше арифметичних обчислень, а також всі обчислення проводяться паралельно, що дає значний виграш у часі.

ВИСНОВКИ

В ході дипломної роботи було досліджено, проаналізовано та порівняно методи обробки, вилучення інформації та енкодування аудіо, сигналів, а також архітектури нейронних мереж та способи класифікації звукових сигналів.

Методи вилучення функції було підібрано та скомбіновано таким чином, щоб максимізувати корисну інформацію та мінімізувати шум та побічні сигнали, також було враховано той факт, що кількість інформації не має бути надлишковою, задля того, щоб уникнути перенавчання алгоритмів.

Метою роботи було не лише дослідити алгоритми машинного навчання, а підібрати такий, що дав би максимальний результат на практиці, а не лише в теорії, а також був стійких до розширення та збільшення інформації, а також добре узагальнював. В якості алгоритмів машинного навчання було використано різні архітектури нейронних мереж для роботи зі звуковими сигналами.

Було проаналізовано два типи нейронних мереж, а саме згорткові та рекурентні НМ, було порівняно якість класифікації, час тренування мереж, стійкість до перенавчання а також шум градієнтного спуску при навчанні мереж.

Кожна з архітектур має свої переваги та недоліки, а саме, рекурентна модель тренується помітно довше, в часовому еквіваленті кожна епоха відпрацьовує мінімум вдвічі довше, також важливим показником є те, що рекурентна мережа дала нижчі результати на тестувальній вибірці, а саме на 1-3% нижче ніж згорткова. По загальноприйнятим метрикам якості, можна рахувати що CNN краще впоралась з поставленою задачею та дала кращі результати на тестуванні, але неочевидним є той факт, що функція втрат згорткової мережі дещо більш зашумлена, це говорить про те, що

градієнтний спуск працював не так ‘плавно’ як у рекурентній мережі, на практиці це може відобразитись на стійкості моделі до узагальнення.

Підсумовуючи, можна зробити висновок, що кожна з типів нейронних мереж має свої очевидні та неочевидні недоліки та переваги, тому найкращим рішенням буде вибір архітектури, залежно від практичної задачі та якості вхідних даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks., 2018.
2. J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences, 1984.
3. Felix Gers. Long Short-Term Memory in Recurrent Neural Networks. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001
4. Richardson, W.: Stacked denoising autoencoders: from morphogenesis to image processing. IEEE Transactions on Image Processing, 5(7):1137–1149, 1996.
5. Hojjat Salehinejad, Julianne Baarbe Understanding and evaluating blind deconvolution algorithms., (2009)
6. Levin, A., Weiss, Y., Durand, F., Freeman, W.T.: Handling outliers in non-blind image deconvolution. In: ICCV. (2011)
7. A. Ya. Grigorenko and S. N. Yaremchenko Analysis of the stress–strain state of inhomogeneous hollow cylinders. International Applied Mechanics, Vol. 52, No. 4, July, 2016.
8. Alex Sherstinsky and Rosalind W. Picard. M-lattice: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11 (2010)
9. Ю.С. Донцова. Анализ методов бинарной класификации / Ю.С. Донцова // Известия Самарского научного центра Российской академии наук. – 2014. – Т. 16. – №6(2). – С. 434-436.
10. David Barber. Bayesian Reasoning and Machine Learning / David Barber. – 2017. – 735 p.
11. Mike Potel. MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java / Mike Potel // Taligent Inc. – 1996. – P. 1-14.
12. В.Е.Гмурман. Руководство к решению задач по теории вероятностей и математической статистике / В.Гмурман. – М.: Высш. школа, 1979. – 400 с.